External Transaction Logic: reasoning and executing transactions involving external domains [1]

Ana Sofia Gomes *, José Júlio Alferes
CENTRIA – Departamento de Informática, Universidade Nova de Lisboa, 2829-516 Caparica, Portugal
Email: sofia.gomes@campus.fct.unl.pt ; jja@fct.unl.pt

Abstract. In this work we present External Transaction Logic, a logic that extends Transaction logic with the ability to model and execute transactions requiring interactions with external entities, as e.g. external web-source, web-services or agents.

Transactions are defined in a logic programming style by the composition of internal and external primitives. These primitives are incorporated in a quite general manner, as a parameter of the External Transaction Logic theory, allowing the specification of transactions integrating knowledge and actions from multiple sources and semantics.

Since one has different control over internal and external domains, different transaction properties are ensured depending on where actions are executed. Namely, internal actions executed in a knowledge base that we fully control, follow the standard model of transactions, where the failure of the transaction leaves the knowledge unaffected. On the other hand, transactional properties over actions executed externally need to be relaxed, as it is impossible to roll back actions executed in a domain that is *external*. To deal with this, external actions can be defined along with compensating operations. If a transaction fails after executing some external action, then these compensations are executed in a backward order to achieve a relaxed model of atomicity.

We provide a model theory for External Transaction Logic, that can be used to reason about the conditions of execution of transactions that require the issuing of both internal and external actions on abstract knowledge bases with potentially different state semantics. We also present here a corresponding proof theory (sound and complete w.r.t. the model theory) that provides means to execute such transactions in a top-down manner.

Keywords: Transaction Logic, updates, external actions, compensations

1. Introduction and Motivation

Several semantics exist to represent and reason about knowledge, each with different meaning, expressivity and complexity depending on the domain for which these semantics have been designed. Additionally, normally such semantics are not static in the sense that they also need to deal with the problem of evolving its knowledge base (KB) by means of updates and actions.

Independently of the semantics chosen for representing the knowledge base of a given application, and of the possible update primitives that can be defined to deal with changes in that KB, just considering the isolated execution of single update primitives is usually not enough. In fact, many situations require the atomic execution of sequences of updates, in an all-or-nothing manner. For example, if one wants to perform a transfer between bank accounts, one must withdraw the amount to transfer from a given account (something that can be done by an update primitive in the bank's KB) and then deposit it into the other account

[1] Some of the results in this paper were preliminary presented in [27] and [28]. This work was partially supported by project ERRO (PTDC/EIA-CCO/121823/2010).
* Author supported by the FCT grant SFRH/BD/64038/2009

(again done by an update primitive). In this case, the *transaction* of a bank transfer is defined by the sequencial execution of those two update actions, in an atomic way.

Transaction Logic ($\mathcal{TR}$) is an extension of predicate logic proposed in [6] to specify and reason about the execution of transactions *independently* of the state and update semantics adopted. $\mathcal{TR}$'s reasoning is supported by a model-theory that allows the study of general properties as equivalence and implication of transactions. Additionally, execution is provided by a proof theory that, being sound and complete w.r.t. the semantics, can answer practical questions like "can this transaction be executed in this state" or "how does my database evolve if this atomic transaction is executed".

In order to achieve its flexibility, both $\mathcal{TR}$'s model and proof theory are parameterized by a pair of oracles defining the semantics of elementary operations (like "insert(p)" or "delete(p)") that query and update the knowledge base. This allows $\mathcal{TR}$ to reason and execute transactions according to a wide variety of query and state change semantics, such as relational databases, well-founded semantics, first-order logic or other non-standard semantics. These characteristics make $\mathcal{TR}$ a powerful tool for reasoning about actions [51], argumentation theories [20], AI planning [6], workflow management and Semantic Web services [52], databases [9], and general KB representation [5].

Example 1 ($\mathcal{TR}$ Financial Transactions)**.** *As an illustration of $\mathcal{TR}$, consider a knowledge base of a bank (taken from [6]) defined by a transactional database and where the balance of a bank account is given by the relation $balance(Acnt, Amt)$. To modify this relation, we are provided with a pair of elementary update operations: $balance(Acnt, Amt).del$ to delete a tuple from the relation, and $balance(Acnt, Amt).ins$ to insert a tuple into the relation. Using these two update primitives, we define four possible transactions: $changeBalance(Acnt, Bal, Bal')$ to change the balance of an account; $withdraw(Amt, Acnt)$ to withdraw an amount from an account in case the amount to transfer is greater than the account's balance; $deposit(Amt, Acnt)$ to deposit an amount into an account, and finally, $transfer(Amt, Acnt, Acnt')$ to transfer an amount from one account to another. These transactions can be defined in $\mathcal{TR}$ in a logic programming style by the following four rules and where the operator $\otimes$, serial conjunction, denotes the sequencial execution of transactions.*

$$
\begin{aligned}
&transfer(Amt, Acnt, Acnt') \leftarrow \\
&\quad withdraw(Amt, Acnt) \otimes deposit(Amt, Acnt') \\
&withdraw(Amt, Acnt) \leftarrow balance(Acnt, B) \otimes \\
&\quad B \geq Amt \otimes changeBalance(Acnt, B, B - Amt) \\
&deposit(Amt, Acnt) \leftarrow balance(Acnt, B) \otimes \\
&\quad changeBalance(Acnt, B, B + Amt) \\
&changeBalance(Acnt, B, B') \leftarrow \\
&\quad balance(Acnt, B).del \otimes balance(Acnt, B').ins
\end{aligned}
$$

Intuitively, the first rule states that a transfer of amount Amt from account $Acnt$ to account $Acnt'$ is performed if first a withdrawal of Amt from $Acnt$ is performed, and then a deposit of the same amount to $Acnt'$ is performed. The last rule states that changing the balance of account $Acnt$ from B to B' is true (in a sequence of knowledge base states) in case first the truth of $balance(Acnt, B)$ is deleted from the knowledge base according to the update-oracle, and then $balance(Acnt, B')$ is inserted.

Many logics systems in the literature of reasoning about actions are limited to reasoning about what formulas are true in a KB (as e.g. in [24,1,23]), or about the direct and indirect effects resulting from a given (trans)action in a KB (as e.g. in [25,36,42,56]). A key feature of $\mathcal{TR}$ is that, besides being able to reason about actions (as e.g. shown in [5,51]), it can also talk about *how* a transaction can be executed in a KB. That is, based on a fixed semantics of states and a fixed semantics of updates (given as a parameter, by oracle instantiations), $\mathcal{TR}$'s semantics can determine what are the paths that allow a transaction to succeed.

More precisely, $\mathcal{TR}$'s semantics deals with statements of the form $P, \langle D_1, D_2, ..., D_{n-1}, D_n \rangle \models t$ meaning that: transaction t succeeds in program P when executed in the state D_1 by changing the system into state D_n through the *path*: $D_1, D_2, ..., D_{n-1}, D_n$. For instance, in the example above, $P, \langle d_1, d_2, d_3 \rangle \models transfer(10, a1, a2)$ holds, if d_1 is e.g. a state where $balance(ac1, 20)$ and $balance(ac2, 30)$ are true, d_2 is a state obtained from d_1 by deleting the truth of $balance(ac1, 20)$ and adding $balance(ac1, 10)$, and d_3 is obtained from d_2 by deleting the truth of $balance(ac2, 30)$ and adding $balance(ac2, 40)$. In fact, in program P, according to $\mathcal{TR}$, the transaction $transfer(10, a1, a2)$ is entailed in any path where in the initial state the balance is at least 10, and where the balances are changed accordingly, and not entailed in any other path. In other words, the paths that entail the transaction are exactly those that correspond to its

complete (atomic) execution. Note that, though in the example above each defined transaction has just one rule, in general $\mathcal{TR}$ allows us to define several rules for a given transaction, making it possible to also deal with non-deterministic actions and transactions. In that case, several different paths may succeed (entail) a transaction, each corresponding to a non-deterministic choice.

However, by just being able to talk about paths where the execution of a transaction completely (and atomically) succeeds, $\mathcal{TR}$ cannot model situations where the failed attempts to execute a transaction cannot be disregarded.

Example 2 (Weekend)**.** *As a very simple example, consider a transaction for preparing my next weekend, where I have 3 possibilities: either going to Paris, to London, or stay at home. To go to either Paris or London, I have to make a reservation for the corresponding travel, this comprising both the booking of a hotel room and flight. These transactions can be written (in a very simplified way) in a $\mathcal{TR}$-like form as follows:*

$$\begin{array}{l} weekendPrep(paris) \leftarrow bookTravel(paris, w). \\ weekendPrep(london) \leftarrow bookTravel(london, w). \\ weekendPrep(home). \\ bookTravel(City, D) \leftarrow \\ \quad findHotel(City, D, H) \otimes findFlight(City, D, F) \\ \quad \otimes reserveHotel(H, D) \otimes reserveFlight(F, D) \end{array}$$

where $reserveHotel(H, D)$ is an action performed externally, e.g. by introducing a tuple corresponding to the reservation in an external KB about hotels, and similarly for flights.

For the sake of this example, imagine that, for that weekend, there are no available hotels in Paris, and no available flights to London (i.e., the corresponding transactions for London and Paris are not entailed in any path). According to $\mathcal{TR}$, in this situation there is a single path entailing the transaction for my weekend preparation, in which I end up staying at home, and nothing changes. Clearly, there is nothing wrong with the success of this transaction in the path where nothing changes. The problem is how, in practice, can a system come up with this only path for succeeding the transaction. Since there are several possible ways to prove $weekendPrep(X)$, any practical system could have tried the several alternatives that in the end fail to be entailed, until a correct one is possibly found.

When considering changes just in an internal KB, as is the case of $\mathcal{TR}$, this does not cause a problem. In fact, since one has full control over its own internal KB, in practice it should be possible to roll back the to internal KB state just before that transaction was tried and failed, and then to try another alternative execution available.

However, when changes are done in external environments, that is no longer possible. If when trying to find a path for succeeding the transaction of preparing my weekend, the transaction $bookTravel(london, w)$ fails because there is no flight available, then something must be done about the previously reserved hotel, before another possibility is tried. But since the KB taking care of hotel reservations is external, rolling back might not even be an option. For example, a money penalty may be associated for canceling a room reservation, in which case the rollback of $reserveHotel(H, D)$ could not simply be the deletion of the added tuple (and which, in principle, one does not have permission to change directly).

While in this travel example, all the actions are external, in general an interaction interleaving internal and external actions is required.

Example 3 (Product request)**.** *Consider now a KB of an organization, that stores information about customers, sales, etc. Moreover, we want the organization to interact with other organizations, customers, suppliers, via web-services, or even by prompting external users to provide information. For example, we may want to define a transaction of satisfying a customer's request for an amount of a product. Such a transaction could (again, in a quite simplified way) be expressed in a $\mathcal{TR}$-like form as follows:*

$$\begin{array}{l} request(Prd, N, Cust) \leftarrow decreaseStock(Prd, N) \\ \quad \otimes dispatch(Prd, N, Cust) \end{array}$$

where $decreaseStock(Prd, N)$ is an internal update of decreasing the stock of product Prd by N (failing when N is greater than the current stock), and $dispatch(Prd, N, Cust)$ is the action of dispatching N units of product Prd to customer $Cust$. In this case, if the dispatch action fails then, before any other possibility to satisfy the request is attempted, the update of decreasing the stock must be rolled back.

Now consider that, another way the organization has to satisfy the request is by asking an associated company whether it has the product, asking the customer whether she accepts that the product is supplied by that other company, and requesting the company to send it to the customer:

$$\begin{array}{l} request(Prd, N, Cust) \leftarrow askComp(Prd, N) \\ \quad \otimes askCust(Cust, Prd) \otimes requestDisp(Prd, N, Cust) \end{array}$$

Here, if the action of asking the customer fails (e.g. because she does not accept it), then unlike the update of decreasing the stock, the action of asking the company cannot be rolled back. Note that, this action can have long lasting effects on the associated company (e.g. by reserving the product). But, since the organization does not control the KB of the associated company, all it can do is to signal that the customer did not accept it. Of course, this would have to be coded in the transaction, and in our proposal it is done by replacing $askComp(Prd, N)$ *in the rule by e.g.* $\mathbf{ext}(askComp(Prd, N), forget(Prod, N))$.

Putting these two rules together, one would expect the transaction to succeed in case the associated company has the product, the customer accepts it, and the product is dispatched by the associate, or by decreasing the stock and dispatching the product. Moreover, the transaction should also succeed in a path where the associate is asked, the customer does not accept the change, the associate is notified to forget about it, and finally by decreasing the stock and dispatching the product. This latter path, that considers the possibility of failed attempts to execute a transaction, can never be obtained by $\mathcal{TR}$.

Consider another example requiring more elaborate KBs, and that is explored further in the remainder of this paper:

Example 4 (Diagnosis example)**.** *Imagine the scenario of an agent with the goal to help in the triage process of an emergency room. For that, the agent's internal KB is defined by an ontology comprising medical information about diseases, medication and so on. Externally, the agent needs to interact with the patient: check her temperature for fever, heart rate, blood pressure, etc. and eventually give medication for her condition. If the agent is able to infer the treatment to be performed and give the patient some medication, then the patient is put in the low priority list.*

However, every medication can have adverse side-effects that, when present, need to be addressed immediately. If that is the case, then the internal information about the patient's priority must change, and something must be given to the patient to counter such side-effects.

The previous examples motivate that, in contrast to what happens in $\mathcal{TR}$, a logic for specifying and reasoning about transactions in an internal KB together with an external environment, cannot ignore the failed attempts to execute transactions, whenever such attempts involved the execution of external actions.

In this sense, the idea of what to do to restore consistency whenever an attempt to execute a transaction fails, corresponds to the notion of *compensation*, proposed originally in the database literature for long-running transactions [22]. Whenever rolling back is an impediment, the solution is to define compensating operations for each external action to be executed. If each compensation reverts the effects of the original action, by executing these compensations in backward order, we obtain an external state considered equivalent to the initial one, achieving a relaxed model of atomicity externally.

In this work we propose $\mathcal{ETR}$, an extension of $\mathcal{TR}$ to reason and execute transactions on knowledge bases partitioned between an internal KB and an external environment. This ability to execute external actions together with internal updates is $\mathcal{ETR}$'s main innovation, and what is missing in the original $\mathcal{TR}$. While actions performed in the internal KB can always be rolled back, actions executed externally need to follow a relaxed model of atomicity, which in $\mathcal{ETR}$'s case is based on compensations.

As in $\mathcal{TR}$, $\mathcal{ETR}$ requires the existence of oracles defining the semantics of the internal KB. Moreover, there must also exist an oracle defining the behavior of the external environment. This allows $\mathcal{ETR}$ to reason and execute transactions that require interaction with external sources without committing to any semantics for the external environment. By instantiating this external oracle with e.g. a Description Logics [1] semantics, or with logics for dynamic external domains like Action Languages [25] or Event Calculus [36], $\mathcal{ETR}$ becomes suitable for a wide range of scenarios like multi-agent systems or the Semantic Web.

In the following, we start by overviewing Transaction Logic's theory (Section 2) which will be used for our contributions. Then we formalize External Transaction Logic (Section 3) by extending $\mathcal{TR}$'s theory to deal with failed paths and compensating operations. For that we define $\mathcal{ETR}$'s syntax (Section 3.1), oracles (Section 3.2), model theory (Section 3.3) and executional entailment (Section 3.4), proving the equivalence to $\mathcal{TR}$ whenever external actions are absent. Afterwards we construct an SLD-style proof theory for a Horn-like subset of the logic that is sound and complete w.r.t. the model theory (Section 3.5). Then we elaborate on the definition of oracles for a Semantic Web context (Section 4), namely Description Logics oracles for both the internal and external KB, and the dynamic description languages – Action Languages, Situation Calculus and Event Calculus – for describ-

ing the external domain. We end with a discussion of related work (Section 5) and conclusions (Section 6). To not disrupt the reading flow, all the proofs of our results are presented as appendix.

2. Background: Transaction Logic

Before introducing External Transaction Logic, we first provide an overview on the $\mathcal{TR}$ framework, including its model and proof theory.

We start by presenting $\mathcal{TR}$'s syntax. For that, without loss of generality (cf. [8]), we work with a Herbrand instantiation of the language as defined in [8]. As usual, the Herbrand universe $\mathcal{U}$ is the set of all ground first-order terms that can be constructed from the function symbols in the language $\mathcal{L}$; the Herbrand base $\mathcal{B}$ is the set of all ground atoms in the language; and a classical Herbrand structure is any subset of $\mathcal{B}$.

To build complex logical formulas, $\mathcal{TR}$ uses the classical logic connectives $\wedge, \vee, \neg, \rightarrow$ and a new connective $\otimes$, denoted *serial conjunction* operator. Informally, the formula $\phi \otimes \psi$ represents an action composed of an execution of ϕ followed by an execution of ψ. Additionally, $\phi \wedge \psi$ defines the action of executing simultaneously ϕ and ψ; while $\phi \vee \psi$ defines the nondeterministic choice of either executing ϕ, ψ or both simultaneously. Finally $\phi \leftarrow \psi$ says that one way to satisfy the execution of ϕ is by executing ψ. Then a $\mathcal{TR}$ program is a set of rules of the form $h \leftarrow \phi$, where h is an atom of the language and ϕ is any complex formula.

A key feature of $\mathcal{TR}$ is the separation of elementary operations from the logic of combining them. With this goal, $\mathcal{TR}$'s theory is parametric on two different oracles allowing the incorporation of a wide variety KB semantics (from classical to non-monotonic to various other non-standard logics). These oracles abstract the representation of KB states and how to query them (data oracle $\mathcal{O}^d$); but also abstract the way states change (transition oracle $\mathcal{O}^t$).

As a result, the language of primitive queries and actions is not fixed, and neither is the definition of a state. Consequently, to distinguish between states, $\mathcal{TR}$ works with a set of state identifiers, each uniquely identifying a state. The *data oracle* $\mathcal{O}^d$ is a mapping from state identifiers to sets of formulas. Intuitively, given a state identifier i, $\mathcal{O}^d(i)$ retrieves the set of formulas true in state i. The *state transition oracle* $\mathcal{O}^t(i_1, i_2)$ is a function that maps pairs of KB states into sets of ground atoms called elementary transitions. Intuitively, given two state identifiers i_1 and i_2, $\mathcal{O}^t(i_1, i_2)$ retrieves the set of elementary transitions that make the KB change from state i_1 into i_2.

The data and transition oracle are strongly related. Particularly, the state identifiers of these two oracles are defined in the same domain. Next we present some examples of data and transition oracles taken from [6].

Relational Oracles A state identifier D is a set of ground atomic formulas. The data oracle simply returns all these formulas, i.e., $\mathcal{O}^d(D) = D$. Moreover, for each predicate symbol p in D, the transition oracle defines two new predicates, $p.ins$ and $p.del$, representing the insertion and deletion of single atoms, respectively. Formally, $p.ins \in \mathcal{O}^t(D_1, D_2)$ iff $D_2 = D_1 + \{p\}$. Likewise, $p.del \in \mathcal{O}^t(D_1, D_2)$ iff $D_2 = D_1 - \{p\}$. SQL-style bulk updates can also be defined by the transition oracle [8] as primitives for creating new constant symbols.

Well-Founded Oracle A state id D is a set of generalized Horn rules[1] and $\mathcal{O}^d(D)$ is the set of literals in the well-founded model of D. Such oracles can represent any rule-base with well-founded semantics, which includes Horn rule-bases, stratified rule-bases, and locally-stratified rule-bases. For advanced applications, one may want to augment $\mathcal{O}^d(D)$ with rules in D. The transition oracle provides primitives for adding and deleting clauses to/from states.

Generalized-Horn Oracle A state D is a set of generalized Horn rules and $\mathcal{O}^d(D)$ is a classical Herbrand model of D. Such oracles can represent Horn rule-bases, stratified rule-bases, locally-stratified rule-bases, rule-bases with stable-model semantics, or any rule-base whose meaning is given by a classical Herbrand model. Again, one may want to augment $\mathcal{O}^d(D)$ with rules in D.

Note that although in the previous examples, state identifiers are defined by sets of formulas, nothing prevents a state identifier to be a set of natural numbers, or some non-logical objects like an XML file. Since a state identifier uniquely identifies a state, from this moment onwards we use the terms of "state" and "state identifier" interchangeably.

2.1. $\mathcal{TR}$ Model Theory

Since the goal of $\mathcal{TR}$ is to account for state changes and the execution of actions on abstract knowledge

[1] Generalized Horn rules are rules with possibly negated premisses

bases, satisfaction is not related to what formulas hold in what states (as this is encapsulated by the oracles) but rather on how actions can be executed in a transactional way. Thus, satisfaction of $\mathcal{TR}$ formulas means *execution*: a formula is said to be true if it can be executed successfully. As a result, contrary to most logics of state change, formulas are not evaluated on states but on *paths*, i.e., sequence of states of the form $\langle D_1, \ldots, D_n \rangle$, where each D_i represents a state. A formula is said to be satisfied in a path if that path is a valid execution trace for that formula.

As most logics, $\mathcal{TR}$ model theory is based on interpretations. An interpretation determines what atoms are true on what paths by defining mappings from paths to Herbrand structures. If $\phi \in M(\pi)$ then, in the interpretation M, path π is a valid execution for the formula ϕ. Additionally, interpretations need to be compliant with the specified oracles. The oracles define elementary primitives which all interpretations must comply to. By limiting the set of possible interpretations to satisfy these restrictions, we can enforce the satisfaction of primitive formulas on the paths that the oracles define it so.

Definition 1 (Interpretations)**.** *An interpretation is a mapping M assigning a classical Herbrand structure (or $\top$[2]) to every path. This mapping is subject to the following restrictions, for all states D_i and every formula φ:*

1. $\varphi \in M(\langle D \rangle)$ *if* $\mathcal{O}^d(D) \models \varphi$
2. $\varphi \in M(\langle D_1, D_2 \rangle)$ *if* $\mathcal{O}^t(D_1, D_2) \models \varphi$

For defining satisfaction of complex formulas over paths, we have first to introduce some operations on paths. For example, the formula $\phi \otimes \psi$ is true (i.e., successfully executes) in a path that executes ϕ up to some point in the middle, and executes ψ from then onwards. To deal with this:

Definition 2 (Path Splits)**.** *A* split *of a path $\pi = \langle S_1, \ldots, S_k \rangle$ is any pair of subpaths, π_1 and π_2, such that $\pi_1 = \langle S_1, \ldots, S_i \rangle$ and $\pi_2 = \langle S_i, \ldots, S_k \rangle$ for some i $(1 \leq i \leq k)$. In this case, we write $\pi = \pi_1 \circ \pi_2$.*

Building on path splits and interpretations, we can now define the general satisfaction of formulas in $\mathcal{TR}$ as follows.

[2]For not having to consider partial mappings, besides formulas, an interpretation can also return the special symbol $\top$. The interested reader is referred to [8] for details.

Definition 3 ($\mathcal{TR}$ Satisfaction of Formulas)**.** *Let M be an interpretation, π a path and ϕ a formula. If $M(\pi) = \top$ then $M, \pi \models \phi$; otherwise:*

1. ***Base Case:*** *$M, \pi \models \phi$ iff $\phi \in M(\pi)$ for any atom ϕ*
2. ***Negation:*** *$M, \pi \models \neg\phi$ iff it is not the case that $M, \pi \models \phi$*
3. ***"Classical" Conjunction:*** *$M, \pi \models \phi \wedge \psi$ iff $M, \pi \models \phi$ and $M, \pi \models \psi$.*
4. ***Serial Conjunction:*** *$M, \pi \models \phi \otimes \psi$ iff $M, \pi_1 \models \phi$ and $M, \pi_2 \models \psi$ for some split $\pi_1 \circ \pi_2$ of path π.*

In the sequel we also mention the satisfaction of disjunctions and implications, where as usual $\phi \vee \psi$ means $\neg(\neg\phi \wedge \neg\psi)$, and $\phi \leftarrow \psi$ means $\phi \vee \neg\psi$.

Example 5 ($\mathcal{TR}$'s Model Theory)**.** *Assume a Relational Database Oracle as defined previously. Since every interpretation M needs to be compliant with the oracles then for every M, $a.ins \in M(\langle \{\}, \{a\} \rangle)$ and $b.ins \in M(\{a\}, \{a, b\})$. Moreover, given the above definition of satisfaction, it is also the case that for every interpretation M, for example, it holds that $M, \langle \{\}, \{a\} \rangle \models a.ins$ and that $M, \langle \{\}, \{a\}, \{a, b\} \rangle \models a.ins \otimes b.ins$.*

2.2. $\mathcal{TR}$ Logical Entailment

After defining how interpretations satisfy complex formulas given a path, we now define which of these interpretations model a formula and a program.

Definition 4 (Models)**.** *An interpretation M is a model of a $\mathcal{TR}$ formula ϕ if $M, \pi \models \phi$ for every path π. In this case, we write $M \models \phi$. An interpretation is a model of a set of formulas if it is a model of every formula in the set.*

This notion of models is mostly used together with the notion of program. Here, a program is a set of formulas of the form $h \leftarrow \phi$ where h, the rule's head, is an atom in the language and ϕ, the rule's body, is any complex formula. Since rules are just complex formulas, an interpretation M models a program P iff it models all its rules, where M models a rule $h \leftarrow \phi$ iff, for every path π, whenever M satisfies the body ϕ it also satisfies the head h (or, equivalently, M either satisfies the head h or it does not satisfy the body ϕ).

Example 6. *Assume a Relational Database Oracle as defined previously, and the following program P.*

$$\mathbf{P}: \quad \begin{array}{l} p \leftarrow a.ins \\ q \leftarrow b.ins \\ q \leftarrow c.ins \\ t \leftarrow p \otimes q \end{array}$$

For every interpretation M that models P it is true that $M, \langle\{\}, \{a\}\rangle \models p$ and $M, \langle\{\}, \{a\}, \{a, b\}\rangle \models t$. Moreover, it is also true that $M, \langle\{\}, \{a\}, \{a, c\}\rangle \models t$

Based on this notion of models, it is also possible to define the notion of entailment in the usual way.

Definition 5 (Logical Entailment)**.** *Let ϕ and ψ be two $\mathcal{TR}$ formulas. Then ϕ entails ψ if every model of ϕ is also a model of ψ. In this case we write $\phi \models \psi$.*

2.3. Executional Entailment and Proof Theory

Besides the concept of a model of a $\mathcal{TR}$ theory, which allows one to prove properties of the theory independently of the paths chosen, $\mathcal{TR}$ also defines the notion of executional entailment. A transaction is entailed by a theory given an initial state, if there is a path starting in that state on which the transaction succeeds. As such, given a transaction and an initial state, the executional entailment determines the path that the KB should follow in order to succeed the transaction. Non-deterministic transactions are possible, an in this case several successful paths exist.

This notion is formalized as follows.

Definition 6 (Executional Entailment)**.** *Let P be a program, ϕ be a formula and $\langle S_1, \ldots, S_n\rangle$ be a path:*

$$P, \langle S_1, \ldots, S_n\rangle \models \phi \tag{1}$$

holds if $M, \langle S_1, \ldots, S_n\rangle \models \phi$ for every model M of P. We write $P, S_1 - \models \phi$ when there exists a path $\langle S_1, \ldots, S_n\rangle$ for which (1) holds.

Example 7. *Recall example 6. Here:*

$$\begin{array}{l} P, \langle\{\}, \{a\}\rangle \models p \\ P, \langle\{a\}, \{a, b\}\rangle \models q \\ P, \langle\{\}, \{a\}, \{a, b\}\rangle \models t \end{array}$$

This latter entailment intuitively states that, given the specifications in P, transaction t successfully executes over a path where initially the KB is empty, then it has the fact a, and finally the facts a and b.

One can also check that:

$$\begin{array}{l} P, \langle\{a\}, \{a, c\}\rangle \models q \\ P, \langle\{\}, \{a\}, \{a, c\}\rangle \models t \end{array}$$

making t and q non-deterministic transactions.

Based on this, $\mathcal{TR}$ defines a proof theory sound and complete w.r.t. executional entailment, and a corresponding implementation to a special class of $\mathcal{TR}$ theories denoted serial-Horn programs [4]. A serial-Horn program P is a set of serial-Horn rules of the form $h \leftarrow b_1 \otimes \ldots \otimes b_n$ where every b_i is an atom and $n \geq 0$.

This proof theory shares some similarities with the SLD-Resolution proof strategy for logic programs. Its goal is to construct a path that corresponds to a valid execution of formula G, i.e., a path $\langle D_0, D_1, \ldots, D_n\rangle$ for which $P, \langle D_0, D_2, \ldots, D_n\rangle \models G$ holds.

This derivation is parametric to the database and transition oracles, $\mathcal{O}^d$ and $\mathcal{O}^t$ which provide the semantics for querying and updating a given state D.

Definition 7 (Proof Theory for $\mathcal{TR}$ Programs)**.** *Let P be a $\mathcal{TR}$ serial-Horn program and D, D_0, D_1, D_2 states. Let G a serial-Horn goal of the form $b_1 \otimes \ldots b_k$ (where every b_i is an atom and $k \geq 0$. The procedure deals with* sequents *of the form $P, D- \vdash G$. The special propositional constant $()$ expresses a tautological formula that is true in every path of length 1.*

A derivation $P \cup \{G\}$ consists of a finite or infinite sequence of sequents $seq_1, seq_2, \ldots, seq_n$ where $seq_1 = P, D_0 - \vdash G$ and each seq_i is either an axiom sequent or is derived from the earlier sequents by the following rules.

Axioms: $P, D- \vdash ()$

Inference Rules: *Let a be an atomic formula, and ϕ and $rest$ be serial goals.*

1. Applying transaction definitions*:*
   Let $a \leftarrow \phi$ be a rule in P, then

$$\frac{P, D- \vdash \phi \otimes rest}{P, D- \vdash a \otimes rest}$$

2. Querying the knowledge base:
   Let $\mathcal{O}^d(D) \models a$, then

$$\frac{P, D- \vdash rest}{P, D- \vdash a \otimes rest}$$

3. Performing elementary updates:
Let $\mathcal{O}^d(D_1, D_2) \models a$*, then*

$$\frac{P, D_2 - \vdash rest}{P, D_1 - \vdash a \otimes rest}$$

Based on this, for a given goal G *an executional derivation (or proof) is said to be* successful *if the sequence* $seq_1, seq_2, \ldots, seq_n$ *is finite and ends in the axiom sequent* $()$*. In this case, we write:*

$$P, \langle D_0, D_1, \ldots, D_n \rangle \vdash G$$

where $D_0, D_1, \ldots, D_n$ *corresponds to the sequence of states appearing respectively in every sequent of the derivation.*

Theorem 1 (Soundness and Completeness [6])**.** *Let* G *be a serial-Horn goal,* $D_0, D_1, \ldots, D_n$ *a path and* P *a serial-Horn program. Then* $P, \langle D_0, D_1, \ldots, D_n \rangle \vdash G$ *iff* $P, \langle D_0, D_1, \ldots, D_n \rangle \models G$

3. Extending Transaction Logic with External Actions

The previously defined Transaction Logic is not suitable for situations where a transaction needs (besides other things) to execute actions in an external domain. In order to better grasp the problem, consider the following example.

Example 8. *Assume the following* $\mathcal{TR}$ *program* P *where* $external\_a$*,* $external\_b$ *and* $external\_c$ *are actions performed externally.*

$$t \leftarrow p.ins \otimes external\_a \otimes external\_b$$
$$t \leftarrow q.ins \otimes external\_c$$

In $\mathcal{TR}$*, transaction* t *has two non-deterministic ways to succeed: if the insertion of predicate* p *followed by the actions* $external\_a$ *and* $external\_b$ *succeed; or if the insertion of predicate* q *followed by* $external\_c$ *succeed.*

However, let's assume that $external\_b$ *fails after the execution of* $external\_a$*. Then,* t *is only satisfied in the path where* $external\_c$ *is performed after* $q.ins$ *(2nd rule).*

Yet, the 1st rule defines an alternative way for executing t *and thus, it can be non-deterministic selected as a legitimate try to succeed it (in* $\mathcal{TR}$*'s proof theory). If this is the case, the actions* $p.ins$ *and* $external\_a$ *are meant to be rolled back (in the implementation version of the proof theory procedure), and this execution is considered to have never happened. However, since* $external\_a$ *corresponds to an external action (e.g. a request to a web-service) it may simply be impossible to roll back such action. Nevertheless, succeeding* t *in that state may still be possible. For that we need to compensate for* $external\_a$ *(e.g. send a message to cancel the previous request), roll back* $p.ins$ *and then execute the 2nd rule.*

The previous example shows an important characteristic of $\mathcal{TR}$'s model and proof theory: it only retrieves the paths where a formula completely succeeds without failures. Particularly, Definition 7 says that a path for transaction t exists if we can construct a proof by making the "right" choices non-deterministically.

Of course, when building proofs (and also, when executing transactions), one cannot assume that the (non-deterministic) choice is always the "right" one. But this causes no problems when executing actions over an internal KB (such as, e.g., a database), because we have a complete control over the KB. I.e., because we can assume that it is always possible to restore any state before any execution is tried. Thus, from an implementation perspective, whenever the system makes a choice that leads to a non-successful derivation, then it simply rolls back to the state previous to that choice, and tries to succeed in an alternative branch. From the proof theory perspective, this execution, where a rollback is performed, is equivalent to the one where the right path was chosen directly.

However, when dealing with external environments and external actions, this is no longer the case. Since it is impossible to roll back an external state, the alternative is to compensate for the external actions already executed, and then succeed in an alternative branch. Contrary to a simple rollback, such execution is not equivalent to choosing the right path directly as it requires an additional interaction with the external world that needs to be reflected in the final path.

Next we propose $\mathcal{ETR}$, an extension of $\mathcal{TR}$ to model such behavior about external domains. For that, $\mathcal{ETR}$'s model theory provides three satisfaction relations for an interpretation M.

Classical Satisfaction Equivalent to $\mathcal{TR}$'s satisfaction of formulas but integrating paths with an external component. $M, \pi \models_c \phi$ if ϕ can execute in π without failures.

Partial Satisfaction Provides the first ingredient to define failures. $M, \pi \models_p \phi$ if either ϕ succeeds

without failures (i.e., if $M, \pi \models_c \phi$) or if it fails because a primitive action in ϕ cannot be executed in a given state.

General Satisfaction Corresponds to the real satisfaction of formulas making use of the two previous notions. $M, \pi \models \phi$ if ϕ succeeds classically over path π or; if we can split π into $\pi = \pi_1 \circ \pi_2$ such that ϕ fails and recovers from this failure in π_1 (by rolling back internally and compensating externally) and succeeds in π_2.

The first two satisfaction relations represent the building blocks for defining failures and are *not* used to satisfy formulas directly. As it shall be precisely defined, a formula ϕ is said to fail in a path π if ϕ can be partially satisfied but not classically satisfied (i.e., if $M, \pi \not\models_c \phi$ but $M, \pi \models_p \phi$). If this is the case, then recovery is in order. For that we need to roll back internally and compensate externally. This is encoded in $M, \pi \rightsquigarrow \phi$ meaning that π is a recovery path obtained after failing to execute ϕ and executing actions externally.

Similarly to what is done in $\mathcal{TR}$, the theory of $\mathcal{ETR}$ has an additional parameter - the external oracle $\mathcal{O}^e$ - defining the states and operations in the external domain. This makes $\mathcal{ETR}$ flexible enough to be used in a wide range of external domains. Transactions are then defined by the composition of internal and external actions in a logic-programming style, allowing the specification of programs that integrate knowledge and actions from multiple sources and semantics.

In this section, we define $\mathcal{ETR}$. We start by introducing $\mathcal{ETR}$'s syntax and external oracle (Sections 3.1 and 3.2). Then we define $\mathcal{ETR}$'s model theory and executional entailment by providing precise meaning for these relations (Sections 3.3 and 3.4). And finally, we introduce a proof procedure that we prove to be sound and complete w.r.t. $\mathcal{ETR}$'s semantics (Section 3.5).

3.1. $\mathcal{ETR}$ Syntax

To deal with external environments and external actions, $\mathcal{ETR}$ operates over a KB including both an internal and an external component. For that, formally $\mathcal{ETR}$ works over two disjoint propositional languages: $\mathcal{L}_P$ (program language), and $\mathcal{L}_\mathcal{O}$ (oracles primitives language). Propositions in $\mathcal{L}_P$ denote actions and fluents that can be defined in the program. As usual, fluents are propositions that can be evaluated without changing the state and actions are propositions that cause evolution of states. Propositions in $\mathcal{L}_\mathcal{O}$ define the primitive actions and queries that deal with the internal and external KB. $\mathcal{L}_\mathcal{O}$ can still be partitioned into $\mathcal{L}_i$ and $\mathcal{L}_a$, where $\mathcal{L}_i$ denotes primitives that query and change the internal KB, while $\mathcal{L}_a$ defines the external actions primitives that can be executed externally. For convenience, it is assumed that $\mathcal{L}_a$ contains two distinct actions `failop` and `nop`, respectively defining trivial failure in the external domain, and trivial success in the external domain without changing the external state.

One of the novelties in $\mathcal{ETR}$ is the retrieval of some paths where a formula is *not* successful (but where some external action was executed and now needs to be compensated). To be better explained below, the difficulty of this notion is that there are several paths where a formula may fail, and not all of them correspond to a valid execution try. To be able to precisely deal with this, we restrict the language of $\mathcal{ETR}$ w.r.t. negation. More concretely, in our restricted language negation can only be applied to atoms:

Definition 8 ($\mathcal{ETR}$ Atoms, Literals, and Formulas)**.** *An $\mathcal{ETR}$* atom *is either a proposition in $\mathcal{L}_P$, $\mathcal{L}_i$ or an external atom. An external atom is either a proposition in $\mathcal{L}_a$ or $\mathbf{ext}(a, b_1 \otimes \ldots \otimes b_j)$ where $a, b_i \in \mathcal{L}_a$. An $\mathcal{ETR}$ literal is either ϕ or $\neg\phi$ where ϕ is an $\mathcal{ETR}$ atom. An $\mathcal{ETR}$* formula *is either a literal, or an expression, defined inductively, of the form $\phi \wedge \psi$, $\phi \vee \psi$ or $\phi \otimes \psi$, where ϕ and ψ are $\mathcal{ETR}$ formulas. We say that a formula is positive iff all its literals are atoms.*

With this definition, external actions can appear in a program in two different ways: 1) without any kind of associated compensation, i.e., $\mathbf{ext}(a, \mathtt{nop})$, and in this case we also write it as $\mathbf{ext}(a)$ or simply a, where $a \in \mathcal{L}_a$ and; 2) with a user defined compensation, written $\mathbf{ext}(a, b_1 \otimes \ldots \otimes b_j)$ where $a, b_i \in \mathcal{L}_a$ $(1 \leq i \leq j)$. The former case should be used when defining external actions that do not need to be compensated, as for instance, external queries. Consequently, if $\mathbf{ext}(a, \mathtt{nop})$ is executed but something fails afterwards, its compensation (i.e., `nop`) trivially succeeds without changing the external state. On the contrary, to define external actions that always fail to be compensated one should use the construct $\mathbf{ext}(a, \mathtt{failop})$. To be able to refer to the external actions in formulas, we define $\mathcal{L}_a^*$ as the augmentation of $\mathcal{L}_a$ with the formulas $\mathbf{ext}(a, b_1 \otimes \ldots \otimes b_j)$ where $a, b_i \in \mathcal{L}_a$.

With the restriction that we impose on the use of negation, we are no longer able to have rules of the

form $\phi \leftarrow \psi$ expressed simply as $\phi \vee \neg\psi$[3], as it is done in $\mathcal{TR}$. However, we still want to be able to express this kind of statements in programs since, as the usage of $\mathcal{TR}$ shows, there are instrumental for defining transactions. Not having rules would surely be a too severe restriction of the language. As such, we define programs as sets of rules of that form, where the head is always an atom from $\mathcal{L}_P$ and the body is any $\mathcal{ETR}$ formula, and then take especial care when defining satisfaction of rules.

Definition 9 ($\mathcal{ETR}$ Programs)**.** *An $\mathcal{ETR}$ program is a set of rules of the form $\phi \leftarrow \psi$ where ϕ is a proposition in $\mathcal{L}_P$ and ψ is an $\mathcal{ETR}$ formula.*

Before we advance to the definition of the semantics of $\mathcal{ETR}$, we give an example to illustrate what kind of transactions can be specified with this language in a domain where these transactions act both on an internal KB and on an external environment.

Example 9. *Recall example 4 from the Introduction where an agent has the task to triage patients in emergency rooms. To do so, the agent needs to interact with the patient, perform a small diagnosis, and assign a priority accordingly. The diagnosis and the priority decision is done internally, using the agent's internal KB defined by a Description Logic. Using this KB, the agent can identify simple cases of flu, and if so, provide treatment to the patient. If this is the case, the agent can give Phenylephrine (PLP) as a treatment. However, this treatment is not suitable for pregnant women or for people suffering from hypertension.*

To encode this, imagine that we have a TBox which includes the following assertions (among many others) related to the Flu and the PLP medicine:

$$\begin{array}{l}
\text{Flu} \sqsubseteq \text{Fever} \sqcap \text{Headache} \sqcap \text{StuffyNose} \sqcap \neg\text{Serious} \\
\text{PLPeligible} \sqsubseteq \neg\text{Pregnant} \\
\text{PLPeligible} \sqsubseteq \neg\text{Hypertense} \\
\text{StrongFever} \sqsubseteq \text{Serious} \\
\text{HeartFailure} \sqsubseteq \text{Serious}
\end{array}$$

To respectively query and update this DL, the agent has the primitives `dlquery`() *and* `dladd`()*. A triage has three possible outcomes: green g, yellow y and red r, respectively ranging from less to high priority. Based on this, the process of triage of a given patient X can be encoded in $\mathcal{ETR}$ by the following rules:*

$$\begin{array}{l}
triage(X, r) \leftarrow diagnosis(X) \otimes \texttt{dlquery}(\text{Serious(X)}) \\
\quad \otimes \texttt{dlquery}(\text{HeartFailure(X)}) \\
triage(X, y) \leftarrow diagnosis(X) \otimes \texttt{dlquery}(\text{Serious(X)}) \\
\quad \otimes \texttt{dlquery}(\neg\text{HeartFailure(X)}) \\
triage(X, g) \leftarrow diagnosis(X) \otimes \texttt{dlquery}(\neg\text{Serious(X)}) \\
triage(X, g) \leftarrow diagnosis(X) \otimes \texttt{dlquery}(\text{Flu(X)})) \\
\quad \otimes \texttt{dlquery}((\text{PLPeligible(X)})) \\
\quad \otimes \mathbf{ext}(giveMeds(X, plp), giveMeds(X, cplp)) \\
\quad \otimes statsOK(X) \\
statsOK(X) \leftarrow diagnosis(X) \otimes \texttt{dlquery}(\neg\text{Serious(X)})
\end{array}$$

In these (very simplified) rules it is stated that if we conclude that the patient's condition is serious and that she suffers from a heart failure, then she must be treated immediately, and thus her priority is defined as red. (1st rule). However, if her condition is serious but she does not show heart failure signs, then the patient's priority is defined as yellow (2nd rule). If the patient's condition is not serious then she is given the green priority (3rd rule). Additionally, if we can conclude that the patient has the flu, and is eligible to receive the treatment, then the agent can give the patient some PLP medication (4th rule). However, if this medication is given, then the agent should ensure that the patient does not become worse afterwards. This is tested by $statsOK$ that re-performs the diagnosis and checks if the status of the patient has become serious (e.g. with the appearance of a strong fever or a heart failure). If this is the case, the call $statsOK$ will fail and the agent will give the patient a medicine to counter the effects of PLP (cplp). Then, depending on the patient displaying symptoms of heart failure or not, the agent will employ the first or the second rule to assign the patient a higher priority. The expression $\mathbf{ext}(giveMeds(X, plp), giveMeds(X, cplp))$ defines an external action with compensation. In this case it states to give patient X the medication plp but, if something fails afterwards, to give $cplp$ as a compensation.

The agent assigns each value according to the results of the diagnosis. A diagnosis corresponds to a battery of tests to check the patient's condition. To perform it, the agent executes external actions to measure (query) the patient's stats. As queries, these actions do not have compensations and thus have the form $\mathbf{ext}(temperature(X, Y))$ (which retrieves the temperature Y of patient X). A simplified version of a diagnosis can be encoded as follows:

[3]Since ψ is defined as a complex formula.

$$
\begin{aligned}
&diagnose(X) \leftarrow checkTemp(X) \otimes checkHeadache(X) \\
&\quad \otimes \ldots \otimes checkHeartRate(X) \\
&checkTemp(X) \leftarrow \mathbf{ext}(temperature(X,Y)) \\
&\quad \otimes[(37 < Y < 41 \otimes \texttt{dladd}(\text{Fever(X)}))\vee \\
&\quad (Y >= 41 \otimes \texttt{dladd}(\text{StrongFever(X)}))\vee \\
&\quad (Y < 37 \otimes \texttt{dladd}(\neg\text{Fever(X)}))] \\
&checkHeadache(X) \leftarrow \mathbf{ext}(hasHeadache(X,Y))\otimes \\
&\quad [(Y = \texttt{true} \otimes \texttt{dladd}(\text{Headache(X)}))\vee \\
&\quad (Y = \texttt{false} \otimes \texttt{dladd}(\neg\text{Headache(X)}))]
\end{aligned}
$$

Note that the compensations for external action are stated directly in the program. In this sense, it is the programmer's responsibility to state the right compensation for each case. This is necessary if the semantics of the external environment is left opened. I.e., if we assume nothing about the semantics of states and updates in the external environment, then it is impossible to automatically infer what are the right actions to repair the effects of a particular action in a given KB.

However, if the semantics of the external environment is known, and formally defined in some logical language, then it may be possible to automatically infer from the external semantics what are the correct compensations for a given action. Although the domains where such notions of automatic compensations can be applied are out-of-scope of this paper, this notion was further developed in [28] and we refer the interested reader to such work for additional details.

We continue by explaining how this external oracle is incorporated in $\mathcal{ETR}$'s theory.

3.2. $\mathcal{ETR}$ External States, and External Oracle

As in $\mathcal{TR}$, both the language and the semantics of $\mathcal{ETR}$ are parameterized by a set of oracles to reason about basic actions and queries. Consequently, besides the data oracle $\mathcal{O}^d$ and the transition oracle $\mathcal{O}^t$ that define the meaning of the internal KB, $\mathcal{ETR}$ integrates an additional external oracle $\mathcal{O}^e$ to evaluate elementary external operations and to abstract the semantics of external states.

As before, states are simply defined by state identifiers. Since $\mathcal{ETR}$ is meant to operate on both an internal KB and external environment, two disjoint sets of state identifiers are needed: one for internal states, and another for uniquely identifying states of the external domain (*external states*). The *external oracle*, $\mathcal{O}^e$, is a mapping from pairs of external states identifiers to formulas in $\mathcal{L}_a^*$. If $\mathcal{O}^e(E_1, E_2) \models \varphi$ then the primitive external action φ is said to execute at the state identified by E_1 yielding the state identified by E_2.

Dealing with state identifiers instead of materialized states is of particular importance when considering external domains. In fact, it may be impossible for the internal system to know what does a particular state identifier mean, as e.g., when dealing with web-services as an external domain. To interact with such external domains, all we need to know is the elementary primitives that can be used to perform queries and updates, and abstract the notion of states to state identifiers. As before, in the remainder we use the terms state as state identifier interchangeably.

Since we stipulated that the actions `failop` and `nop` always belong to $\mathcal{L}_a^*$ with a precise meaning, we also enforce that for every external oracle and every pair of external states $\mathcal{O}^e(E_1, E_2) \not\models$ `failop` and $\mathcal{O}^e(E_1, E_1) \models$ `nop` (i.e., `failop` always fails, and `nop` always succeed leaving the state unchanged, as desired).

External actions with compensations, $\mathbf{ext}(a, b_1 \otimes \ldots \otimes b_j)$, are evaluated by the external oracle solely according to what is known about a, i.e., for every oracle we impose that $\mathcal{O}^e(E_1, E_2) \models \mathbf{ext}(a, b_1 \otimes \ldots \otimes b_j)$ iff $\mathcal{O}^e(E_1, E_2) \models a$ for any $a, b_i \in \mathcal{L}_a$. Thus, as expected, it is not the task of the oracle (but rather of the $\mathcal{ETR}$ semantics, as we shall see) to deal with compensations.

Note that, $\mathcal{TR}$ requires *two* oracles, $\mathcal{O}^d$ and $\mathcal{O}^t$, to respectively define the semantics of queries and updates. This separation promotes the distinction between the static semantics of states and its dynamics. However, these two oracles are not independent of each other and they must share the same set of state identifiers. In practice, although this separation may ease the task of implementing these oracles (because of the separation of the two concepts), nothing prevent us from expressing both oracles using a single mapping.

Because of this, in $\mathcal{ETR}$ we assume that only one external oracle is needed to characterize the behavior of the external environment w.r.t. a given primitive (both queries and updates). Since little may be known about the external domain, it may not be possible to distinguish between an external query and an external update and thus, we assume that every external primitive can cause a state transition. If q is a query, then the primitive is mapped to a pair with the same state, i.e., $q \in \mathcal{O}^e(S, S)$.

The external oracle abstracts the theory and semantics of the external domain, encapsulating the elementary operations that can be performed externally. In the

sequel we see this oracle as a black box, that encodes the behavior of the external domain in a completely independent way. However, if one wants to reason about both the internal and the external KB, one can fix $\mathcal{O}^e$ by formalizing it with some logic for describing external worlds. Thus, after defining $\mathcal{ETR}$'s semantics we also elaborate upon the role of this external oracle in Section 4.2 and show how it can be instantiated with Description Logics, Action Languages, Event Calculus or Situation Calculus.

3.3. $\mathcal{ETR}$ Model Theory

As in $\mathcal{TR}$, formulas in $\mathcal{ETR}$ are evaluated on *paths*, i.e., on a sequence of states. Since $\mathcal{ETR}$ deals with an external environment a state S is now a pair (D, E) where D represents an internal state and E denotes an external state. Based on this, a path is a sequence of states. Moreover, for convenience and to help the semantics to deal with recovery of external failures, we also record in paths the operations performed between states, defined as follows.

Definition 10 (States and Paths)**.** *An $\mathcal{ETR}$* state *S is a pair (D, E) where D and E are, respectively, internal and external states. A* path *of length k, or a* k-path*, is a finite sequence of states, $\langle S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{k-1}} S_k \rangle$ where A_is ($1 \leq i < k$) are atoms from $\mathcal{L}_a^*$ or $\mathcal{L}_i$.*

In this definition, a path $\langle S_i \xrightarrow{A_i} S_{i+1} \rangle$ means that action A_i caused the change from state S_i into state S_{i+1}. If A is a formula of the form $\mathbf{ext}(a, b_1 \otimes \ldots \otimes b_j)$ then this allows us to know that $b_1 \otimes \ldots \otimes b_j$ is the compensation to be performed in the event of a failure after the execution of the external action a.

Interpretations are defined in the usual way, but now they also incorporate the external oracle.

Definition 11 (Interpretations)**.** *An interpretation is a mapping M assigning a classical Herbrand structure (or $\top$) to every path. This mapping is subject to the following restrictions, for all states (D_i, E_j) and every atom φ defined in $\mathcal{L}_{\mathcal{O}}$:*

1. *$\varphi \in M(\langle (D, E) \rangle)$ iff $\mathcal{O}^d(D) \models \varphi$ for any external state E*
2. *$\varphi \in M(\langle (D_1, E) \xrightarrow{\varphi} (D_2, E) \rangle)$ iff $\mathcal{O}^t(D_1, D_2) \models \varphi$ for any external state E*
3. *$\varphi \in M(\langle (D, E_1) \xrightarrow{\varphi} (D, E_2) \rangle)$ iff $\mathcal{O}^e(E_1, E_2) \models \varphi$ for any internal state D*

Note that $\mathcal{ETR}$ rules cannot have oracle atoms in their head (cf. Definition 9). I.e., in $\mathcal{ETR}$, contrary to $\mathcal{TR}$, it is not possible to redefine oracle primitives in programs. This restriction makes the logic cleaner by strictly separating between primitives and complex actions defined in a program, without particularly limiting the expressive power of the language. In fact, every $\mathcal{TR}$ program can be re-written to comply with this restriction. Consider for instance that the rule $a.ins \leftarrow c.ins \otimes d.ins$ exists in the program, then we can rewrite it as:

$$new\_a\_ins \leftarrow a.ins$$
$$new\_a\_ins \leftarrow c.ins \otimes d.ins$$

and replace all the remaining occurrences of $a.ins$ in the program by new_a_ins. Due to this design choice, interpretation functions in $\mathcal{ETR}$ restrict oracle atoms to be true only when oracles define it so.

Before defining satisfaction of formulas, besides the operation of path splits, which is simply inherited from $\mathcal{TR}$, we need the additional notion of ending of a path, which will be helpful to isolate the exact point of failure of a transaction.

Definition 12 (Ending of a Path)**.** *The* ending *of a k-path π corresponds to the 1-path $\pi_{\mathtt{end}}$ composed of the last state of π, i.e., if $\pi = \langle S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{k-1}} S_k \rangle$ then $\pi_{\mathtt{end}} = \langle S_k \rangle$.*

$\mathcal{TR}$'s standard definition of satisfaction can now be easily adapted to accept the $\mathcal{ETR}$'s notion of paths, where a state S is a pair consisting of an internal and external state (D, E):

Definition 13 (Classical Satisfaction)**.** *Let M be an interpretation, π a path and ϕ a formula. If $M(\pi) = \top$ then $M, \pi \models_c \phi$; otherwise:*

1. ***Base Case:*** *$M, \pi \models_c \phi$ iff $\phi \in M(\pi)$ for any atom ϕ*
2. ***Negation:*** *$M, \pi \models_c \neg\phi$ iff it is not the case that $M, \pi \models_c \phi$*
3. ***"Classical" Disjunction:*** *$M, \pi \models_c \phi \vee \psi$ iff $M, \pi \models_c \phi$ or $M, \pi \models_c \psi$.*
4. ***"Classical" Conjunction:*** *$M, \pi \models_c \phi \wedge \psi$ iff $M, \pi \models_c \phi$ and $M, \pi \models_c \psi$.*
5. ***Serial Conjunction:*** *$M, \pi \models_c \phi \otimes \psi$ iff $M, \pi_1 \models_c \phi$ and $M, \pi_2 \models_c \psi$ for some split $\pi_1 \circ \pi_2$ of path π.*

As discussed above, $\mathcal{TR}$'s satisfaction does not consider the possibility of failure. Since $\mathcal{ETR}$ allows external actions as transaction formulas, it must take into the account the possibility of a transaction to fail. Particularly, if a failure occurs after the execution of ex-

ternal actions, then we need to execute some compensating operations to invert the external actions already performed, and recover a consistent state in the external KB. As such, $\mathcal{ETR}$'s model theory also needs to address how the external recovery can be ensured in case of external failures. The partial satisfaction relation below is the first ingredient to deal with such failures.

Example 10 (Running Example)**.** *Recall example 8, but now external expressions like $external\_a$ are replaced by external actions with compensations:*

$$t \leftarrow p.ins \otimes \mathbf{ext}(a, a_1 \otimes a_2) \otimes \mathbf{ext}(b, b_1)$$
$$t \leftarrow q.ins \otimes \mathbf{ext}(c, c_1)$$

Moreover, assume that the internal KB is a relational database formalized as explained in Section 2, and the external oracle includes: $\mathcal{O}^e(e_1, e_2) \models a$, (i.e., the external execution of a in state e_1 succeeds, and makes the external world evolve into e_2), $\mathcal{O}^e(e_1, e_5) \models c$, and that for every state E, $\mathcal{O}^e(e_2, E) \not\models b$ (i.e., the execution of b in state e_2 fails).

In this example, it can be easily checked that the formula $p.ins \otimes \mathbf{ext}(a, a_1 \otimes a_2)$ is classically satisfied in the path:

$$\langle(\{\}, e_1) \xrightarrow{p.ins} (\{p\}, e_1) \xrightarrow{\mathbf{ext}(a, a_1 \otimes a_2)} (\{p\}, e_2)\rangle$$

Similarly, the formula $q.ins \otimes \mathbf{ext}(c, c_1)$ is classically satisfied in the path:

$$\langle(\{\}, e_1) \xrightarrow{q.ins} (\{q\}, e_1) \xrightarrow{\mathbf{ext}(c, c_1)} (\{q\}, e_5)\rangle$$

Furthermore, given the external oracle definition above, it is easy to see that $\mathbf{ext}(b, b_1)$ cannot succeed in any path starting in state e_2.

The idea of partial satisfaction is to identify the path:

$$\langle(\{\}, E_1) \xrightarrow{p.ins} (\{p\}, E_1) \xrightarrow{\mathbf{ext}(a, a_1 \otimes a_2)} (\{p\}, E_2)\rangle$$

as one that satisfies the formula $p.ins \otimes \mathbf{ext}(a, a_1 \otimes a_2) \otimes \mathbf{ext}(b, b_1)$ up to some point, though it eventually fails.

A formula is partially satisfied if it either completely succeeds or if it succeeds up to some point, and then fails in a primitive action. Specifically:

Definition 14 (Partial Satisfaction)**.** *Let M be an interpretation, π a path and ϕ a formula. If $M(\pi) = \top$ then $M, \pi \models_p \phi$; otherwise:*

1. ***Base Case:*** *$M, \pi \models_p \phi$ iff ϕ is an atom and one of the following holds:*
   (a) $M, \pi \models_c \phi$
   (b) $M, \pi \not\models_c \phi$, $\phi \in \mathcal{L}_i$, $\pi = \langle(D, E)\rangle$ *and* $\neg\exists D_i$ *s.t.* $M, \langle(D, E) \xrightarrow{\phi} (D_i, E)\rangle \models_c \phi$
   (c) $M, \pi \not\models_c \phi$, $\phi \in \mathcal{L}^*_a$, $\pi = \langle(D, E)\rangle$ *and* $\neg\exists E_i$ *s.t.* $M, \langle(D, E) \xrightarrow{\phi} (D, E_i)\rangle \models_c \phi$
2. ***Negation:*** *$M, \pi \models_p \neg\phi$ iff it is not the case that $M, \pi \models_p \phi$*
3. ***"Classical" Disjunction:*** *$M, \pi \models_p \phi \vee \psi$ iff $M, \pi \models_p \phi$ or $M, \pi \models_p \psi$*
4. ***"Classical" Conjunction:*** *$M, \pi \models_p \phi \wedge \psi$ iff $M, \pi \models_p \phi$ and $M, \pi \models_p \psi$*
5. ***Serial Conjunction:*** *$M, \pi \models_p \phi \otimes \psi$ iff one of the following holds:*
   (a) $M, \pi \models_p \phi$ *and* $M, \pi \not\models_c \phi$
   (b) $\exists$ *split* $\pi_1 \circ \pi_2$ *of path* π *s.t.* $M, \pi_1 \models_c \phi$ *and* $M, \pi_2 \models_p \psi$

There may be several reasons for a formula not to be classically satisfied in a given path, and, by design, not all of them are taken into consideration in the definition of partial satisfaction. For example, given a relational oracle, the query a, independently of the interpretation M, is not classically satisfied in the path $\langle(\{a\}, E) \xrightarrow{b.ins} (\{a, b\}, E)\rangle$. Similarly, the action $b.ins$ is never satisfied in the 1-path $\langle(\{a\}, E)\rangle$. However, these failures are not interesting in the sense that they do not correspond to a real execution-try. Particularly, $b.ins$ can classically succeed in a path starting in $\langle(\{a\}, E)\rangle$, and the query a is true in the 1-paths composed of any of the singleton states $\langle(\{a\}, E)\rangle$ and $\langle(\{a, b\}, E)\rangle$. As such, partial satisfaction is defined in such a way that, although for any M, $M, \langle(\{a\}, E) \xrightarrow{b.ins} (\{a, b\}, E)\rangle \not\models_c a$ (i.e., a fails), it is also the case that $M, \langle(\{a\}, E) \xrightarrow{b.ins} (\{a, b\}, E)\rangle \not\models_p a$.

In other words, our definition of partial satisfaction only deals with failures that come from a real impediment to executing a primitive action from a particular state S_0. In the case of atomic queries, this means that the given query is not true in a particular 1-path, and in the case of atomic action, it means that there is no possible evolution from S_0 that successfully satisfies the action (items 1b and 1c respectively).

With these definitions of classical and partial satisfaction, intuitively a "legitimate" failure is one where a formula is partially but not classically satisfied in a path. Moreover, cf. claim 1 of Proposition 1 below, the path where this happens must always end exactly in the state prior to the failure. This is the reason why in

definition of partial satisfaction failures of primitives are constrained to 1-paths in items 1b and 1c. These 1-paths represent the state where the transaction failed.

Besides this result, Proposition 1 shows additional properties of the partial satisfaction definition. Claim 2 states that we can weaken a formula that is partially but not classically satisfied using the serial conjunctive operator $\otimes$, i.e., that in any path π where a ϕ can be partially but not classically executed, then $M, \pi \models_p \phi \otimes \psi$ for every formula ψ. Additionally, for positive formulas, the partial satisfaction is a relaxed version of the classical satisfaction (claim 3), and the two satisfaction relations coincide whenever they are evaluating atoms that are not specified by the oracles (claim 4).

Proposition 1. *Let M be an interpretation, π a path, $\pi_{\texttt{end}}$ the 1-path containing the last state of π, ϕ and ψ any $\mathcal{ETR}$ formulas, ϕ' a positive formula, ϕ_P an atom from $\mathcal{L}_P$ and a an atom such that $a \in \mathcal{L}_i$ or $a \in \mathcal{L}_a^*$.*

1. *If $M, \pi \models_p \phi$ and $M, \pi \not\models_c \phi$ then $\exists a$ s.t. a occurs in ϕ, $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$*
2. *If $M, \pi \models_p \phi$ and $M, \pi \not\models_c \phi$ then $M, \pi \models_p \phi \otimes \psi$*
3. *If $M, \pi \models_c \phi'$ then $M, \pi \models_p \phi'$*
4. *$M, \pi \models_c \phi_P$ iff $M, \pi \models_p \phi_P$*

Consider again example 8. In $\mathcal{TR}$, just like in logic programming, we can satisfy t by satisfying either the first body or the second. But, as motivated above, in $\mathcal{ETR}$ we want consider an additional way of satisfying t. Namely, t should also be satisfied if the first body is "tried", compensated, and then the second body is successfully executed. With the definitions above, we made precise what is meant by the "tried", viz. it is partially but not classically satisfied. The next definitions specify what is left, i.e., how to successfully compensate a formula that is partially but not classically satisfied.

However, for this, some additional operations on paths are needed. We start by defining the notion of a *rollback path* of a given path. Intuitively, given a path π, its rollback path is obtained by keeping all externally executed actions, and rolling back on the internal state:

Definition 15 (Rollback Path, and Sequence of External Actions)**.** *Let π be a k-path of the following form $\langle (D_1, E_1) \xrightarrow{A_1} (D_2, E_2) \xrightarrow{A_2} \ldots \xrightarrow{A_{k-1}} (D_k, E_k) \rangle$. The* rollback *path of π is the path obtained from π by:*

1. *Replacing all D_is by D_1*
2. *Keeping just the transitions where $A_i \in \mathcal{L}_a^*$.*

The sequence of external actions of π, denoted $\texttt{Seq}(\pi)$, is the sequence of actions of the form $\mathbf{ext}(a, b_1 \otimes \ldots \otimes b_j)$ that appear in the transitions of the rollback path of π.

Example 11 (Rollback path)**.** *Consider the path $\pi = \langle (\{\}, E_1) \xrightarrow{p.ins} (\{p\}, E_1) \xrightarrow{\mathbf{ext}(a, a_1 \otimes a_2)} (\{p\}, E_2) \rangle$. The* rollback path *$\pi_0$ of π is the path obtained by rolling back the internal state and keeping the external transitions. Thus, $\pi_0 = \langle (\{\}, E_1) \xrightarrow{\mathbf{ext}(a, a_1 \otimes a_2)} (\{\}, E_2) \rangle$.*

Note that the operator $\texttt{Seq}(\pi)$ only collects the external actions that have the form $\mathbf{ext}(a, b_1 \otimes \ldots \otimes b_j)$. Since our aim is to to compensate the executed actions, then actions without compensations are skipped. Alternatively, to define compensations that always fail, one should use the primitive `failop` as in $\mathbf{ext}(a, \texttt{failop})$.

Building on this, we define the notion of a *recovery path*. After rolling back the internal state and retrieving all the necessary compensations, external recovery is achieved by executing the compensation operations defined in $\texttt{Seq}(\pi)$ in the inverse order.

Definition 16 (Inversion, and Recovery Path)**.** *Let $S = \langle \mathbf{ext}(a_1, \mathcal{B}_1), \ldots, \mathbf{ext}(a_n, \mathcal{B}_n) \rangle$ be a sequence of actions from $\mathcal{L}_a^*$, and $\mathcal{B}_i$ is a sequence of actions of the form $(b_{1_i} \otimes \ldots \otimes b_{k_i})$. Then, the inversion of S is the transaction formula $\texttt{Inv}(S) = \mathcal{B}_n \otimes \ldots \otimes \mathcal{B}_1$.*

π_r is a recovery path *of $\texttt{Seq}(\pi)$ w.r.t. M iff $M, \pi_r \models_c \texttt{Inv}(\texttt{Seq}(\pi))$.*

Example 12 (Rollback and Recovery)**.** *Recall example 10 and further assume the following for the external oracle: $\mathcal{O}^e(e_2, e_3) \models a_1$ and $\mathcal{O}^e(e_3, e_4) \models a_2$.*

From example 11 we know that the rollback path π_0 of $\pi = \langle (\{\}, E_1) \xrightarrow{p.ins} (\{p\}, E_1) \xrightarrow{\mathbf{ext}(a, a_1 \otimes a_2)} (\{p\}, E_2) \rangle$ is $\pi_0 = \langle (\{\}, E_1) \xrightarrow{\mathbf{ext}(a, a_1 \otimes a_2)} (\{\}, E_2) \rangle$.

Thus, by Definition 16, $\texttt{Seq}(\pi_0) = \langle \mathbf{ext}(a, a_1 \otimes a_2) \rangle$ and $\texttt{Inv}(\texttt{Seq}(\pi_0)) = a_1 \otimes a_2$. Finally, given our previous stated external oracle definitions, we know that:

$$\langle (\{\}, E_2) \xrightarrow{a_1} (\{\}, E_3) \xrightarrow{a_2} (\{\}, E_4) \rangle$$

is a recovery path of $\texttt{Seq}(\pi)$ w.r.t. any interpretation M.

Equipped with these auxiliary definitions, we can finally make precise what we mean by compensating a formula that is partially but not classically satisfied. To this end, $M, \pi \rightsquigarrow \phi$ states that given an interpretation M, the path π is a path where all external actions is-

sued due to the execution of formula ϕ are compensated, and the internal state is rolled back.

Definition 17 (Compensating Path for a Transaction)**.** *Let M be an interpretation, π a k-path ($k \geq 2$) with external actions in the transitions, and ϕ a formula. $M, \pi \rightsquigarrow \phi$ if* all *the following conditions are true:*

1. *$\exists \pi_1$ such that $M, \pi_1 \models_p \phi$ and $M, \pi_1 \not\models_c \phi$*
2. *$\exists \pi_0$ such that π_0 is the rollback path of π_1*
3. *$\mathtt{Seq}(\pi_0) \neq \emptyset$ and $\exists \pi_r$ such that π_r is a recovery path of $\mathtt{Seq}(\pi_0)$ w.r.t. M*
4. *π_0 and π_r are a split of π, i.e., $\pi = \pi_0 \circ \pi_r$*

Example 13 (Compensating Path)**.** *In the scenario of the previous examples 10–12, the statement*

$$M, \langle\, (\{\}, e_1)^{\mathbf{ext}(a, a_1 \otimes a_2)} \rightarrow (\{\}, e_2)^{a_1} \rightarrow (\{\}, e_3)^{a_2} \rightarrow (\{\}, E_4) \rangle \rightsquigarrow p.ins \otimes \mathbf{ext}(a, a_1 \otimes a_2) \otimes \mathbf{ext}(b, b_1)$$

holds for any interpretation M. Note that this path does not satisfy the formula (since $\mathbf{ext}(b, b_1)$ fails). Instead, it leaves the internal and external KBs in a state somehow equivalent to the initial state: the operations done in the internal KB are rolled back, and the externally executed actions are compensated.

A compensating path for a formula ϕ is one where ϕ is *not* successfully executed, but where external recovery can still be guaranteed. Also note that they are only defined for cases where, besides a primitive action fails, some external actions with compensations were executed. This is so, because the operator $\mathtt{Seq}(\pi_0)$ only collects external actions of the form $\mathbf{ext}(a, b_1 \otimes \ldots \otimes b_j)$. This is as expected: if no external actions were executed in π_0 or if all the external actions executed are not meant to be compensated (e.g. if they are external queries), then $\mathtt{Seq}(\pi_0) = \emptyset$. Intuitively, if this is the case then no compensations are needed, and the formula just fails (as in standard $\mathcal{TR}$).

Based on these definitions, we are finally able to formalize which (complex) formulas are true on which paths.

Definition 18 (General Satisfaction)**.** *Let M be an interpretation, π a path and ϕ a formula. If $M(\pi) = \top$ then $M, \pi \models \phi$; otherwise:*

1. ***Base Case:*** *$M, \pi \models \phi$ if $\phi \in M(\pi)$ for any atom ϕ*
2. ***Negation:*** *$M, \pi \models \neg\phi$ if it is not the case that $M, \pi \models \phi$*
3. ***"Classical" Disjunction:*** *$M, \pi \models \phi \vee \psi$ if $M, \pi \models \phi$ or $M, \pi \models \psi$.*
4. ***"Classical" Conjunction:*** *$M, \pi \models \phi \wedge \psi$ if $M, \pi \models \phi$ and $M, \pi \models \psi$.*
5. ***Serial Conjunction:*** *$M, \pi \models \phi \otimes \psi$ if $M, \pi_1 \models \phi$ and $M, \pi_2 \models \psi$ for some split $\pi_1 \circ \pi_2$ of π.*
6. ***Compensating Case:*** *$M, \pi \models \phi$ if $M, \pi_1 \rightsquigarrow \phi$ and $M, \pi_2 \models \phi$ for some split $\pi_1 \circ \pi_2$ of π*
7. *For no other M, π and ϕ, $M, \pi \models \phi$.*

This definition strongly resembles Definition 13. Intuitively, with this general notion of satisfaction, a formula ϕ succeeds if it succeeds classically, or if although a primitive action failed to be executed, the system can recover from the failure and ϕ can still succeed in an alternative path (item 6). As expected, recovery only makes sense in situations where some external actions were performed before the failure. Otherwise we can just roll back to the initial state and try to satisfy the formula in an alternative branching.

Example 14. *Recall examples 10–12, and further assume $\mathcal{O}^e(e_4, e_5) \models c$. Based on this, the complex formula: $(p.ins \otimes \mathbf{ext}(a, a_1 \otimes a_2) \otimes \mathbf{ext}(b, b_1)) \vee (q.ins \otimes \mathbf{ext}(c, c^{-1}))$ is satisfied both in the path*

$$\langle (\{\}, E_1)^{q.ins} \rightarrow (\{q\}, E_1)^{\mathbf{ext}(c, c_1)} \rightarrow (\{q\}, E_5) \rangle$$

(without using compensations), and also in the path

$$\langle (\{\}, E_1)\ ^{\mathbf{ext}(a, a_1 \otimes a_2)} \rightarrow (\{\}, E_2)^{a_1} \rightarrow (\{\}, E_3)^{a_2} \rightarrow (\{\}, E_4)^{q.ins} \rightarrow (\{q\}, E_4)^{\mathbf{ext}(c, c_1)} \rightarrow (\{q\}, E_5) \rangle$$

by using item 6 (where the first disjunct i.e. tried, fails and is compensated).

As previously stated, the general satisfaction is strongly related with the classical satisfaction. Particularly, besides the compensating case, the definition of general satisfaction exactly coincides with classical satisfaction (Definition 13). We state this correspondence as follows.

Theorem 2. *Let M be an interpretation, ϕ any formula, and ϕ' a positive formula and π, π' paths such that π' is a path where no external actions appear in the transitions. Then:*

$$\text{If } M, \pi \models_c \phi' \text{ then } M, \pi \models \phi' \tag{2}$$

$$M, \pi' \models_c \phi \text{ iff } M, \pi' \models \phi \tag{3}$$

Models and logical entailment can now be defined as usual:

Definition 19 (Model and entailment of a formula)**.** *An interpretation M is a model of a formula ϕ (denoted $M \models \phi$) iff $M, \pi \models \phi$ for every path π.*
A formula ϕ logically entails another formula ψ ($\phi \models \psi$) if every model of ϕ is also a model of ψ.

Since $\mathcal{ETR}$ restricts the use of negation to atoms, we have to explicitly define what it means to model a rule and to model a program. Intuitively, an interpretation models a rule if, whenever it models its body, it also models its head; as usual, it models a program if it models all its rules. Moreover, to deal with compensations, we further impose that, for models of rules, compensating paths and classical satisfaction of the rule body correspond with the compensating paths and classical satisfaction for the head:

Definition 20 (Model of a Program)**.** *An interpretation M models a rule $head \leftarrow body$ iff for every path π:*

- *If $M, \pi \models body$ then $M, \pi \models head$ and;*
- *If $M, \pi \models_c body$ then $M, \pi \models_c head$ and;*
- *If $M, \pi \rightsquigarrow body$ then $M, \pi \rightsquigarrow head$*

An interpretation M is a model of a program P if it models all its rules. In this case we write $M \models P$.

A program P entails another program P' ($P \models P'$) if all models of P are models of P'. Two programs P and P' are equivalent iff $P \models P'$ and $P' \models P$.

3.4. Executional Entailment

Logical entailment, be it of formulas or programs, takes into account all the possible execution paths of a transaction formula. Hence, this entailment can be used to define general equivalence and implication of formulas, as one can express properties like "whenever transaction ϕ is executed, ψ is also executed" ($\phi \models \psi$) or "transaction ϕ is equivalent to transaction ψ" ($\phi \models \psi$ and $\psi \models \phi$); or of programs, as one can specify that a program is equivalent to another program (if they entail one another).

Useful as this might be, sometimes one needs a simpler kind of reasoning that is concerned only with a particular execution of a formula. As such, similarly to $\mathcal{TR}$, in addition to logical entailment $\mathcal{ETR}$ supports another entailment called *executional entailment*. Whilst logical entailment allows one to *reason* about $\mathcal{ETR}$ theories, executional entailment provides a logical account of *execution* of $\mathcal{ETR}$.

Definition 21 (Executional Entailment)**.** *Let P be a program, ϕ be a formula and $\langle S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{n-1}} S_n \rangle$ be a path. The statement:*

$$P, \langle S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{n-1}} S_n \rangle \models \phi \tag{4}$$

is true if $M, \langle S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{n-1}} S_n \rangle \models \phi$ for every model M of P. We write $P, S_1 - \models \phi$ when there exists a path $S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{n-1}} S_n$ that makes (4) true.

$P, \langle S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{n-1}} S_n \rangle \models \phi$ means that, given a program P, the path $\langle S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{n-1}} S_n \rangle$ represents a valid execution for transaction ϕ.

Example 15. *In our running example, both statements below hold:*

$\bullet P, \langle (\{\}, e_1) \xrightarrow{\mathbf{ext}(a, a_1 \otimes a_2)} (\{\}, e_2) \xrightarrow{a_1} (\{\}, e_3) \xrightarrow{a_2} (\{\}, e_4) \xrightarrow{q.ins} (\{q\}, e_4) \xrightarrow{\mathbf{ext}(c, c_1)} (\{q\}, e_5) \rangle \models t$
$\bullet P, \langle (\{\}, e_1) \xrightarrow{q.ins} (\{q\}, e_1) \xrightarrow{\mathbf{ext}(c, c_1)} (\{q\}, e_5) \rangle \models t$

The latter correspond to executing, and succeeding, t by the second rule. The former amounts to trying to execute t by the first rule, failing, rolling back and compensating, and then executing and succeeding t by the second rule.

$P, S_1 - \models \phi$ accounts for situations where all one wants to know is whether ϕ can succeed starting from state S_1 under P, e.g. $P, (\{\}, e_1) - \models t$ (meaning that t succeeds if executed in that initial state).

As expected, $\mathcal{ETR}$ is a conservative extension of $\mathcal{TR}$. Namely, if ϕ and P are valid in both logics (e.g. do not contain external actions), then the two logics coincide, i.e., they satisfy the same formulas in the same paths. This is encoded in Theorem 3. Obviously, since paths in $\mathcal{ETR}$ have an additional external component than in $\mathcal{TR}$, the paths only coincide in their shared internal path.

Theorem 3 (Relation to $\mathcal{TR}$)**.** *Let P be a transaction program and ϕ a transaction formula such that P and ϕ are both well-formed in $\mathcal{TR}$'s and in $\mathcal{ETR}$'s syntax. Then for any external state E:*

$$P, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E) \rangle \models_{ETR} \phi \text{ iff } P, \langle D_1, \ldots, D_n \rangle \models_{TR} \phi$$

3.5. A Proof Procedure for $\mathcal{ETR}$

Executional entailment determines what is the meaning of executing a transaction defined in a program, starting from an initial state. Our next step is to define a procedure for proving transactions in that way. In

this section we extend the proof theory for the ground[4] serial-Horn $\mathcal{TR}$ fragment as described in Definition 7. The advantage of this fragment is that it can be formulated as a least-fixpoint in a logic programming style.

A serial-Horn program P is a finite set of *serial goals*. A serial goal is a transaction formula of the form $a_1 \otimes a_2 \otimes \ldots \otimes a_n$, where each a_i is an atom and $n \geq 0$. When $n = 0$, we write (), which denotes the empty goal. A *serial-Horn rule* has the form $b \leftarrow a_1 \otimes \ldots \otimes a_n$, where the body $a_1 \otimes \ldots \otimes a_n$ is a serial goal and the head b is an atom.

Here, we present a procedure to verify that $P, S_0^- \models \phi$, i.e, that a transaction ϕ can succeed starting from the state $S_0 = (D_0, E_0)$ and, in case of success, to obtain a path starting in S_0 that satisfies ϕ.

This procedure starts with a program P, an initial state S_0 and a serial goal ϕ and manipulates *resolvents*. At each step the procedure non-determinstically applies a series of rules to the current resolvent until it either reaches the empty goal and succeeds, or no more rules are applicable and fails. Moreover, if the procedure succeeds, it also returns a path in which the goal succeeds. To cater for this last requirement, resolvents contain the information about the path obtained so far. A resolvent is of the form $\pi, S_i \Vdash_P \phi$, meaning that, we want to execute transaction ϕ in P from the initial state S_i; where π records the path history.

With this form of resolvents, a proof, or successful derivation, for $P, S_0^- \models \phi$ starts with $\langle S_0 \rangle, S_0 \Vdash_P \phi$ and applies the rules defined below, until eventually it reaches a resolvent $\pi, S_f \Vdash_P ()$. If such a proof is found, then we further conclude that $P, \pi \models \phi$ (where π starts with S_0 and ends with S_f). I.e., not only we prove that ϕ can succeed starting from S_0, but we also found a path where ϕ succeeds.

The rules for this derivation are specified in Figure 1. Derivation rules $\mathbf{r_1} - \mathbf{r_4}$ are equivalent to the ones in $\mathcal{TR}$'s proof theory. Rule $\mathbf{r_1}$ applies a transaction definition by unfolding its definition rule, i.e., if we are proving a given atom that is defined in the program in the head of a rule, we can replace this atom in the resolvent by the body of that rule. Rule $\mathbf{r_2}$ deals with a query to the oracle: if we are proving an atom defined in the database oracle, and the database oracle satisfies this atom in that particular state, then we can simply remove this atom. Finally, rules $\mathbf{r_3}$ and $\mathbf{r_4}$ respectively define the execution of an internal and external update: if the atom is a primitive action A that can succeed in the current state S_1 leading to state S_2 (i.e., A is true in $\langle S_1 \xrightarrow{A} S_2 \rangle$) then we can remove the atom from the resolvent and update the path and state appropriately. This set of rules forms the basis of the so called $SLD_{\mathcal{ETR}}$ classical derivation.

Moreover, to deal with compensations, $SLD_{\mathcal{ETR}}$ has an additional rule (rule $\mathbf{r_5}$). For these, besides the usual notion of successful derivation, we need to deal with derivations that do not succeed, but that end in the execution of an external action that fails in the oracle. If that is the case (i.e., if rule $\mathbf{r_5}.(a)$ is applicable) , then we need to roll back internal actions (rule $\mathbf{r_5}.(b)$), compensate the executed actions (rule $\mathbf{r_5}.(c)$), and proceed (rule $\mathbf{r_5}.(d)$). To handle derivations that fail we further identify *action-failed* derivations. These correspond to $SLD_{\mathcal{ETR}}$ derivations that end in a resolvent of the form $\pi, S_f \Vdash_P L_1 \otimes \ldots \otimes L_n$ and L_1 is an external action primitive that cannot be executed in S_f.

Definition 22 ($SLD_{\mathcal{ETR}}$ Derivation and Classical Derivation)**.** *An $SLD_{\mathcal{ETR}}$-derivation (resp. classical derivation) for a serial goal ϕ in a program P and state S_0 is finite a sequence of resolvents starting with $\langle S_0 \rangle, S_0 \Vdash_P \phi$, and obtained by non-deterministically applying the rules r_1–r_5 (resp. r_1–r_4) specified in Figure 1.*

Definition 23 (Successful and Action-failed Derivations)**.** *Let P be a program, ϕ a serial goal and S_0 an initial state. An $SLD_{\mathcal{ETR}}$-derivation (resp. classical derivation) for ϕ in P starting in S_0 is* successful *if it ends in a resolvent of the form $\pi, S_f \Vdash_P ()$. In this case we write $P, \pi \vdash \phi$ (resp. $P, \pi \vdash_c \phi$).*
The derivation is action-failed *if it ends in a resolvent of the form $\pi, S_f \Vdash_P L_1 \otimes \ldots \otimes L_n$ s.t.:*

(i). $L_1 \in \mathcal{L}_i$, $\mathcal{O}^d(D_f) \not\models L_1$
and $\neg\exists D_i$ *s.t.* $\mathcal{O}^t(D_f, D_i) \models L_1$, *or*
(ii). $L_1 \in \mathcal{L}_a^*$ *and* $\neg\exists E_i$ *s.t.* $\mathcal{O}^e(E_f, E_i) \models L_1$

Note the parallel between these definitions and the satisfaction relations $\models_c$, $\models_p$ and $\models$. Classical derivation, as well as $\models_c$, does not consider the possibility of failures (and, as such, cannot use rule $\mathbf{r_5}$). Action-failed derivations, can only "fail" (and this failure be considered part of a valid derivation) in case it is impossible to execute a given primitive external action from a particular state, just like in $\models_p$. Finally, similarly to $\models$, an $SLD_{\mathcal{ETR}}$-derivation is either a classical derivation, or it includes some failed derivations from which it can recover by rolling back the internal

[4]The restriction to ground formulas is not essential and can be easily lifted. We only require it in order to simplify the presentation.

Let $\pi, (D_1, E_1) \Vdash_P L_1 \otimes \ldots \otimes L_n$ be a resolvent. Then the next resolvent in the derivation is defined by:

$\mathbf{r_1}$. $\pi, (D_1, E_1) \Vdash_P B_1 \otimes \ldots \otimes B_j \otimes L_2 \otimes \ldots \otimes L_n$ if $L_1 \leftarrow B_1 \otimes \ldots \otimes B_j \in P$
$\mathbf{r_2}$. $\pi, (D_1, E_1) \Vdash_P L_2 \otimes \ldots \otimes L_n$ if $\mathcal{O}^d(D_1) \models L_1$
$\mathbf{r_3}$. $\pi \circ \langle (D_1, E_1) {}^{L_1}\!\!\rightarrow (D_2, E_1)\rangle, (D_2, E_1) \Vdash_P L_2 \otimes \ldots \otimes L_n$ if $\mathcal{O}^t(D_1, D_2) \models L_1$
$\mathbf{r_4}$. $\pi \circ \langle (D_1, E_1) {}^{L_1}\!\!\rightarrow (D_1, E_2)\rangle, (D_1, E_2) \Vdash_P L_2 \otimes \ldots \otimes L_n$ if $\mathcal{O}^e(E_1, E_2) \models L_1$
$\mathbf{r_5}$. $\pi \circ \langle S_1 {}^{A_1}\!\!\rightarrow \ldots {}^{A_{p-1}}\!\!\rightarrow S_p {}^{A_k^{-1}}\!\!\rightarrow \ldots {}^{A_1^{-1}}\!\!\rightarrow S_q\rangle, S_q \Vdash_P L_1 \otimes \ldots \otimes L_n$ if all of the following conditions hold:

(a) There is an action-failed classical derivation starting in $\langle S_1\rangle, S_1 \Vdash_P L_1 \otimes \ldots \otimes L_n$ (where $S_1 = (D_1, E_1)$) ending in $\langle S_1 {}^{A_1}\!\!\rightarrow \ldots {}^{A_{j-1}}\!\!\rightarrow S_j\rangle, S_j \Vdash_P \phi$, for some transaction ϕ
(b) $S_1 {}^{A_1}\!\!\rightarrow \ldots {}^{A_{p-1}}\!\!\rightarrow S_p$ is the rollback path of $S_1 {}^{A_1}\!\!\rightarrow \ldots {}^{A_{j-1}}\!\!\rightarrow S_j$ (cf. Definition 15)
(c) $\texttt{Inv}(\texttt{Seq}(\langle S_1 {}^{A_1}\!\!\rightarrow \ldots {}^{A_{p-1}}\!\!\rightarrow S_p\rangle)) = A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ (cf. Definition 16)
(d) There is successful classical derivation for $\langle S_p\rangle, S_p \Vdash_P A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ ending in $\langle S_p {}^{A_k^{-1}}\!\!\rightarrow \ldots {}^{A_1^{-1}}\!\!\rightarrow S_q\rangle, S_q \Vdash_P ()$

Fig. 1. $SLD_{\mathcal{ETR}}$-derivation rules

state, compensate all the previously executed external actions, and succeed in an alternative path.

Taken together, definitions 22 and 23 determine a sound and complete procedure w.r.t. the semantics to find the paths that satisfy a transaction ϕ given a program P and an initial state S_0. This procedure resembles an SLD-style procedure and can be seen as an extension of the inference system for serial-$\mathcal{TR}$ as presented in [4]. The main differences when compared to $\mathcal{TR}$'s inference system are the evaluation of external actions w.r.t to an external oracle $\mathcal{O}^e$ and the non-deterministic possibility of executing compensations in the derivation.

Example 16 (Proof Theory)**.** *Recall our previous running examples 10-14, and imagine we want to find the proof for $P, \langle(\{\}, E_1)\rangle^- \models t$. With this as goal, we have to start in the resolvent:*

$$\langle(\{\}, E_1)\rangle, (\{\}, E_1) \Vdash_P t \tag{5}$$

In this resolvent, we can apply the rules $\mathbf{r_1}$ (where we unfold t for $p.ins \otimes \mathbf{ext}(a, a_1 \otimes a_2) \otimes \mathbf{ext}(b, b_1)$), $\mathbf{r_3}$ and $\mathbf{r_4}$ respectively, and reach the resolvent:

$$\pi_1, (\{p\}, E_2) \Vdash_P \mathbf{ext}(b, b_1) \tag{6}$$

where $\pi_1 = \langle\ (\{\}, E_1) {}^{p.ins}\!\!\rightarrow (\{p\}, E_1) {}^{\mathbf{ext}(a, a_1 \otimes a_2)}\!\!\rightarrow (\{p\}, E_2)\rangle$

Since $\mathbf{ext}(b, b_1) \in \mathcal{L}_a^*$ and by the external oracle definition, $\neg\exists E_i$ s.t. $\mathcal{O}^e(E_2, E_i) \models b$, then this derivation (from (5) to (6)) is action-failed*. Since we have not used rule $\mathbf{r_5}$, this derivation is also denoted a* classical *action-failed derivation. Additionally, by Definitions 15 and 16, we know that the rollback path of π_1 is the path:*

$$\pi_0 = \langle(\{\}, E_1) {}^{\mathbf{ext}(a, a_1 \otimes a_2)}\!\!\rightarrow (\{\}, E_2)\rangle$$

and that $\texttt{Inv}(\texttt{Seq}(\pi_0)) = a_1 \otimes a_2$*. Moreover, from the resolvent:*

$$\langle(\{\}, E_2)\rangle, (\{\}, E_2) \Vdash_P a_1 \otimes a_2 \tag{7}$$

we can apply the rule $\mathbf{r_4}$ twice obtaining:

$$\pi_2, (\{\}, E_4) \Vdash_P () \tag{8}$$

where $\pi_2 = \langle(\{\}, E_2) {}^{a_1}\!\!\rightarrow (\{\}E_3) {}^{a_2}\!\!\rightarrow (\{\}, E_4)\rangle$

Since we have never applied rule $\mathbf{r_5}$ and reached the empty goal (), this derivation (from (7) to (8)) is a successful classical derivation.

Building on these, we can again start in the resolvent

$$\langle(\{\}, E_1)\rangle, (\{\}, E_1) \Vdash_P t \tag{9}$$

and apply rule $\mathbf{r_5}$, and thus reach the resolvent

$$\pi_0 \circ \pi_2, (\{\}, E_4) \Vdash_P t \tag{10}$$

Afterwards, we can apply rule $\mathbf{r_1}$ (where we unfold t for $q.ins \otimes \mathbf{ext}(c, c_1)$), $\mathbf{r_3}$ and $\mathbf{r_4}$, and reach the resolvent:

$$\pi_f, (\{q\}, E_5) \Vdash_P () \tag{11}$$

where $\pi_f = \pi_0 \circ \pi_2 \circ \langle\ (\{\}, E_4) {}^{q.ins}\!\!\rightarrow (\{q\}, E_4) {}^{\mathbf{ext}(c, c_1)}\!\!\rightarrow (\{q\}, E_5)\rangle$

This latter derivation (from (9) to (11)) is denoted a successful $SLD_{\mathcal{ETR}}$*-derivation, and thus we write* $P, \pi_f \vdash t$.

Theorem 4 (Soundness and Completeness of $\vdash$)**.** *Let P be a serial-Horn program, ϕ a serial-Horn goal, and π be a path starting in state S_0 and ending in S_f. Then, $P, \pi \models \phi$ iff $P, \pi \vdash \phi$.*

As it happens with $\mathcal{TR}$, the implementation of the proof procedure (which is a natural future milestone of this work) is a first step for achieving a system that executes transactions in the real world, according to $\mathcal{ETR}$'s semantics. Obviously, such a system would have to deal with several other issues, such as: having efficient ways to represent the states and the primitive actions, communication with the external environment to execute the external actions and receive feedback of their success or failure, durability of the changes made in the knowledge base, etc.

In $\mathcal{TR}$, if one wants to execute a transaction t, given a program P, and a KB in a state S_0, one can start a derivation for: $P, S_0 - \vdash t$. If this derivation succeeds, then the KB's state is committed to the last state obtained by the proof; if it fails the user is eventually warned of that, and the KB is rolled back to S_0, so as not to leave any trace of the failed attempt to execute the transaction (and guarantee the atomicity property of transactions). In $\mathcal{ETR}$ something similar can be done in case of success of a transaction: simply start an $SLD_{\mathcal{ETR}}$-derivation for $\langle S_0 \rangle, S_0 \Vdash_P t$, and if it succeeds ending in resolvent $\pi, S_f \Vdash_P ()$, just commit to S_f. However, in case of transaction failure, special care must be taken.

If the $SLD_{\mathcal{ETR}}$-derivation fails, then the internal KB can be rolled back to the initial state, just as in $\mathcal{TR}$. Regarding the external environment, inevitably, there will be traces of the failed attempt to execute the transaction. Nevertheless, $\mathcal{ETR}$ and its proof procedure provide all the appropriate mechanisms to issue compensations for the executed external actions and, in case all these compensations succeed, a relaxed model of atomicity is still achieved. As it was previously mentioned, it is the programmer's task to, given her knowledge about the external environment, specify which compensations should be executed for which actions, in each situation. However, in general, compensations can still fail in the external environment, or even be non-existent. In these cases, if a transaction fails, actions executed externally may be left without any compensation. Note that this is as expected, since one may not have enough information about the external environment, and cannot control its external state.

In some cases, the implementation itself can try to minimize the possibility of external failures. For example, the implementation can postpone the execution of external actions as much as possible, for instance by pre-processing the program (using the $\mathcal{ETR}$'s model theory to guarantee that the pre-processing yields an equivalent program) to anticipate the execution of internal actions until it does not depend on external information (required e.g., by variable instantiation).

But ultimately, the external environment may be such that no compensations for actions ever succeed, and in such a case nothing can be done. In fact, one can only *completely* guarantee that all compensations indeed revert the effects of external actions performed, if one knows the exact behavior of the external environment, and the compensations are automatically computed.In [28] we have considered this possibility, and proposed a way for computing the right compensations when the external environment is fully specified using the Action Language $\mathcal{C}$. This way, besides guaranteeing that compensations indeed succeed when a transaction fails, this approach also eases the programmer's burden to specify the correct compensations.

However, note that this can only be achieved if the appropriate oracles for the external environment are fully specified. In the next section, we illustrate how this oracle specification can be done.

4. $\mathcal{ETR}$ Oracles for the Web

Both the semantics and the proof procedure for $\mathcal{ETR}$ are parametric on three oracles: two of them defining the behavior of the internal KB, and a third one defining the behavior of the external environment. To be able to make any practical use of $\mathcal{ETR}$, one has to instantiate each of these oracles in such a way that they model both the behavior of the internal KB and the behavior of the external environment intended for that specific usage.

In this section, for the sake of illustration, we make the exercise of fully describe each of these oracles for a specific internal KB and for some external environments. Most of this illustration is inspired by the possible usage of $\mathcal{ETR}$ in the Semantic Web. The choice of the Semantic Web as application domain, is due to our stance that $\mathcal{ETR}$ is indeed useful in this domain, as is further discussed in Section 6 below.

More precisely, for the exercise in this section, the internal KB is described by a *DL-Lite* database, and the interaction with this database is made by either asking conjunctive queries to the *DL-Lite* database, via a `dlquery()` primitive; or by updating the ABox part of the database, via a `dladd()` primitive.

Regarding the external environment we explore several possibilities that go beyond the approach adopted until now (i.e., the approach where one knows nothing about how the external world behaves, and thus the external oracle is seen as a "black-box"). For a Semantic Web illustration, we first consider the case where the external environment is just another *DL-Lite* database. However, as an *external* database, one can interact with it by querying or attempting to change the ABox, but without having complete control over these interactions. For other application domains, we also consider an external environment that is modeled by one of the following well established languages for describing effects of actions in external domains - Action Languages [25], Situation Calculus [42], and Event Calculus [36]. This list is surely non-exhaustive, and several other oracle formalizations are possible. For example, while here we are only considering external oracle instantiations where the laws stating how actions change the world are immutable, we could also have defined oracle instantiations where the action theory itself can evolve and change, for instance based on the works of [16,57]. This is however left outside the scope of this section.

4.1. Description Logics Oracles

As mentioned above, in this section we formalize the oracle for the case where the internal knowledge base is described by a *DL-Lite* database.

As usual in Description Logic (DL) [1], a knowledge base $\mathcal{K}$ is composed by a TBox ($\mathcal{T}$), defining the concepts and terminologies, and an ABox ($\mathcal{A}$), with assertions of particular instances

We can abstractly define a database oracle $\mathcal{O}^d$ as a mapping from states, which in this case are a DL knowledge base $(\mathcal{T}, \mathcal{A})$, to the set of formulas that are true in it. Different oracle instantiations can be defined for different Description Logics. Here, we have provide an instantiation for the *DL-Lite* Family [11]. *DL-Lite* is the backbone of the OWL-2 QL profile [18] and known for its low computational complexity on large volumes of instance data (ABox size). OWL 2 [46] is the second edition of the standard OWL [43], and has three different profiles with the goal to address different application requirements. The OWL 2 QL profile is designed to deal with very large amounts of data and in contexts where query answering is the most important task.

DL-Lite also defines a family of languages. Here we illustrate the definition of an oracle for *DL-Lite*$_{\mathcal{FR}}$, a language that has polynomial algorithms to update the ABox [26,13]. We start by recalling some definitions from *DL-Lite*$_{\mathcal{FR}}$.

4.1.1. DL-Lite knowledge bases

As any DL language, elementary descriptions are partitioned between atomic *concepts* and atomic *roles*, and complex descriptions can be built from these using concept constructors. To build complex descriptions, *DL-Lite*$_{\mathcal{FR}}$ has the following constructs:

$$\begin{array}{l} B ::= A \mid \exists R \\ C ::= B \mid \neg B \\ R ::= P \mid P^- \end{array}$$

where A denotes an atomic concept, B a basic concept, C a general concept, P an atomic role and R a basic role. Based on these, an ABox $\mathcal{A}$ is a set of membership assertions of the form $B(a)$ and $P(a, b)$ where a and b are object constants; and a TBox $\mathcal{T}$ is a set assertions of the form:

$B \sqsubseteq C$	concept inclusion assertion
$R_1 \sqsubseteq R_2$	role inclusion assertion
$(\texttt{funct}\ R)$	role functionality assertion

The semantics is defined by first-order logic interpretations. An interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ has a non-empty domain Δ, and a mapping $\cdot^{\mathcal{I}}$ from individuals, concepts and roles to Δ as follows:

$$\begin{array}{l} A^{\mathcal{I}} \subseteq \Delta \\ P^{\mathcal{I}} \subseteq \Delta \times \Delta \\ (P^-)^{\mathcal{I}} = \{(a_2, a_1) \mid (a_1, a_2) \in P^{\mathcal{I}}\} \\ (\exists R)^{\mathcal{I}} = \{a \mid \exists a'.(a, a') \in R^{\mathcal{I}}\} \\ (\neg B)^{\mathcal{I}} = \Delta \setminus B^{\mathcal{I}} \end{array}$$

To simplify, we assume *standard names*, i.e., we assume that there is no distinction between the alphabet of constants and Δ. An interpretation $\mathcal{I}$ satisfies a given TBox or ABox assertion F (denoted by $\mathcal{I} \models F$) if the following is true.

- $\mathcal{I} \models D_1 \sqsubseteq D_2$, if $D_1^{\mathcal{I}} \subseteq D_2^{\mathcal{I}}$
- $\mathcal{I} \models (\texttt{funct}\ R)$, if $(a_1, a_2) \in R^{\mathcal{I}}$ and $(a_1, a_3) \in R^{\mathcal{I}}$ implies $a_2 = a_3$

- $\mathcal{I} \models B(a)$, if $a \in B^{\mathcal{I}}$
- $\mathcal{I} \models R(a,b)$, if $(a,b) \in R^{\mathcal{I}}$

An interpretation is a model of a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ (written $\mathcal{I} \models \mathcal{K}$) iff it satisfies all the assertions in $\mathcal{K}$. We say that $\mathcal{K}$ is *satisfiable* if it has at least one model, and *unsatisfiable* otherwise.

4.1.2. DL-Lite database oracle

After specifying what is a *DL-Lite* database, something that correspondes to the $\mathcal{ETR}$ notion of a state, in order to use it in $\mathcal{ETR}$ we need to define what are the primitives that make the interaction between the programs (or $\mathcal{ETR}$ formulas) and the knowledge base. As mentioned before, we consider that the interaction is made by either asking conjunctive queries to the *DL-Lite* database, via a `dlquery`() primitive, or by updating the ABox of the database, via a `dladd`() primitive.

For the `dlquery`() primitive, we can make use of *DL-Lite* characteristics and employ a query-answering algorithm for conjunctive queries as defined in [12]. A conjunctive query $q(\vec{x})$ over the KB $\mathcal{K}$ is an expression of the form:

$$q(\vec{x}) \equiv \exists \vec{y}.conj(\vec{x}, \vec{y})$$

where $\vec{x}$ and $\vec{y}$ are respectively known as the *distinguished variables* and *non-distinguished variables* of the query; and $conj(\vec{x}, \vec{y})$ is a conjunction of atoms of the form $B(z)$ or $R(z_1, z_2)$ where B is a basic concept and R a role in $\mathcal{K}$, and z, z_1, z_2 are either constants in $\mathcal{K}$ or variables in $\vec{x}$ or $\vec{y}$.

The answers to a conjunctive query $q(\vec{x})$ as above in a knowledge base $\mathcal{K}$, denoted by $\texttt{ans}(q(\vec{x}), \mathcal{K})$ (or by $\texttt{ans}(\exists \vec{y}.conj(\vec{c}, \vec{y}), \mathcal{K})$), is the set of tuples $\vec{c} \in \Delta \times \ldots \times \Delta$ such that when the variables $\vec{x}$ are substitute with the constants $\vec{c}$, the formula $\exists \vec{y}.conj(\vec{x}, \vec{y})$ is true in every $\mathcal{I}$ that is a model of $\mathcal{K}$.

For asking conjunctive queries, our primitive takes the form $\texttt{dlquery}(\vec{c}, conj(\vec{c}, \vec{y}))$ where $conj(\vec{c}, \vec{y})$ is a conjunctive query, and $\vec{c}$ the set of constants that appear in it. The database oracle $\mathcal{O}^d$ is:

Definition 24 (*DL-Lite* database oracle)**.** *Let $\mathcal{K}$ be a state (i.e., a TBox and an ABox in DL-Lite).*

$$\mathcal{O}^d(\mathcal{K}) \models \texttt{dlquery}(\vec{c}, conj(\vec{c}, \vec{y})) \textit{ iff } \texttt{ans}(\exists \vec{y}.conj(\vec{c}, \vec{y}), \mathcal{K}) \neq \emptyset$$

Note that, in this definition we are only dealing with boolean DL queries. This is because, from the start, we are working with Herbrand interpretations of the transaction logic formulas. As such, all rules are ground, and so are the DL queries possibly appearing in them. For the general case of rules with variables, a condition similar to DL-safety [48] would have to be imposed, so as to guarantee that the instantiation of the variables in $\mathcal{ETR}$ rules would not depend on the result of the queries. In this context, DL-safety must guarantee for every transaction rule that every variable in $\vec{x}$ of a query is instantiated before the query call. Then for such rule, this implies that every variable in $\vec{x}$ to occur previously (i.e., in a sequence of rule bodies with $\otimes$) in a predicate defined in $\mathcal{L}_P$. Since in this paper we only present the ground version of $\mathcal{ETR}$, and since for that the discussion on safety is not crucial, here we do not elaborate further on this topic.

4.1.3. DL-Lite transition oracle

For the `dladd`() primitive, we restrict it to instance-based updates, i.e., to updates on the membership assertions of the ABox. For those, an update $\mathcal{U}$ is simply a set of ABox assertions that is integrated into the current knowledge base $\mathcal{K}$, obtaining a new knowledge base $\mathcal{K}'$.

However, the updated information may leave $\mathcal{K}'$ unsatisfiable, and, in this case, the conflicts between $\mathcal{U}$ and the old information from $\mathcal{K}$ need to be addressed. Since the resulting knowledge base $\mathcal{K}'$ may not be expressible in the original DL where it was defined [2], solving such conflicts may not be trivial (even for the simpler case of ABox updates).

To address this problem, several formal operators have been proposed either based on models and on formulas updates, and the interested reader is referred to [41] for more details.

For the purpose of this illustration, we assume the Careful Semantics Update as presented in [13]. Nevertheless, note that any other update semantics could be as easy defined (based either on TBox, ABox updates or both).

A careful update is defined for a *DL-Lite*$_{\mathcal{FR}}$ $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ as follows:

$$\texttt{c\_upd}(\mathcal{T}, \mathcal{A}, \mathcal{U}) := \mathcal{A}^c_m \cup \mathcal{U}$$

where $\mathcal{A}^c_m$ is the careful maximal set of assertions subset of the closure of $\mathcal{A}$ w.r.t. $\mathcal{T}$ compatible with $\mathcal{U}$.

Based on this, we finally define `dladd`():

Definition 25 (*DL-Lite* transition oracle)**.** *Let a state S be a pair $\langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is a ABox. Let $\mathcal{U}$ be a set of ABox assertions to update the ABox and* `c_upd` *the careful semantics update algorithm ref-*

erenced earlier.

$$\mathcal{O}^t(\langle\mathcal{T},\mathcal{A}\rangle,\langle\mathcal{T},\mathcal{A}'\rangle) \models \texttt{dladd}(\mathcal{U}) \textit{ iff } \mathcal{A}' = \texttt{c\_upd}(\mathcal{T},\mathcal{A},\mathcal{U})$$

4.2. External oracles

Instantiating in detail the internal oracles is required in order to use $\mathcal{ETR}$ (or $\mathcal{TR}$), and to precisely define what the internal primitives stand for. In contrast, the external oracle can be left practically open, when all we want to do is to execute an $\mathcal{ETR}$ program. This is as expected since, in most cases, it is impossible to know how the external oracle is specified. But, reasoning about general properties of $\mathcal{ETR}$ transactions (like equivalence or implication) can only be done if the behavior of the external environment is known, and in that case an external representation of states must be chosen and the external oracle must be properly defined.

In a Semantic Web context, the external environment can also be described by a Description Logic. This is e.g. the case when we have an ontology distributed across several web-sources, or when a rule system interacts with an ontology on the web (like e.g., in hybrid MKNF knowledge bases [45,47,34]). Consequently, in the following we define an external oracle $\mathcal{O}^e$ for a *DL-Lite* knowledge base similar to what we have done for the internal oracles $\mathcal{O}^d$ and $\mathcal{O}^t$.

However, in general, acting upon an external environment is not restricted to querying or (trying) to add or delete information from external KBs. In fact, acting on external sources (e.g. when interacting via web-services) is in general much richer, and these external sources usually admit a great diversity of actions. Several languages for describing this diversity of actions, have been defined and studied in the literature. To make $\mathcal{ETR}$ fully integrated with external environments whose interaction can be established by this variety of actions, one has to define external oracles for such languages. For that, we also provide a $\mathcal{O}^e$ specification for Action Languages, Situation Calculus, and Event Calculus.

4.2.1. External Description Logic Oracle

If the external environment is described by a Description Logic knowledge base, on which one can freely query and update it, then defining the external oracle is just equivalent to defining the internal oracles $\mathcal{O}^d$ and $\mathcal{O}^t$ for that given DL. The only minor difference is that, since $\mathcal{O}^e$ works with pairs of states even when evaluating queries, querying $\mathcal{O}^e$ is equivalent to the definition of $\mathcal{O}^d(\mathcal{K})$ for a given state $\mathcal{K}$, but where the pair state does not change, i.e., $\mathcal{O}^e(\mathcal{K},\mathcal{K})$.

However, even if the external environment is made up of ontologies described by some Description Logic, it is not common that others can freely query and update them. In this sense, it may be useful to restrict the set of information that can be updated in $\mathcal{O}^e$ by external entities. This is a common feature of several systems where one is required to authenticate in order to haver permission to access and change a given table, tuple or webpage.

To achieve this, we can define in $\mathcal{O}^e$ a permission list, linking users to sets of ABox assertions that can be modified. Then, when updating the KB, we must verify that the update is safe, i.e., that only the permitted assertions are modified. Such an external oracle for a *DL-Lite* DL can be defined as follows:

$$\mathcal{O}^e(\mathcal{K},\mathcal{K}) \models \texttt{dlquery}(\vec{c}, conj(\vec{c},\vec{y})) \text{ iff } \texttt{ans}(\exists\vec{y}.conj(\vec{c},\vec{y}),\mathcal{K}) \neq \emptyset$$

$$\mathcal{O}^e(\langle\mathcal{T},\mathcal{A}\rangle,\langle\mathcal{T},\mathcal{A}'\rangle) \models \texttt{dladd}(\mathcal{U}) \text{ iff } \mathcal{A}' = \texttt{c\_upd}(\mathcal{T},\mathcal{A},\mathcal{U}) \wedge safe(\mathcal{A},\mathcal{A}',list(user))$$

where $safe(\mathcal{A},\mathcal{A}',List(User))$ denotes that every assertion that is present in $\mathcal{A}'$ and not in $\mathcal{A}$ is defined in the allowed list for that user. Clearly, other possibilities could be defined and explored. For example, besides a permission list for atoms that can be updated, one can also have permissions for atoms to be queried, either by a simple list, or with more sophisticated mechanisms. This, however, is beyond the scope of this paper.

4.2.2. Action Languages Oracle

Generally, external environments are much richer than a simple ontology, over which one may query or update, subject to permission. In fact, the external environment normally responds to different kinds of actions, whose effects can be modeled, if one knows enough about the environment. Without any model of the kinds of actions that can be performed in the environment, certainly no reasoning can be done about their effects. Action Languages are a family of languages proposed in [25], exactly with the goal to model the dynamics of external environments, and in this section we show how an oracle for these languages can be defined, in order to combine reasoning about actions performed in an external environment, with the definition of transaction in $\mathcal{ETR}$ that also act on an internal knowledge base.

Every action language defines a series of *laws* describing actions in the world and their effects. Which laws are possible, as well as the syntax and semantics of each law, depends on the action language in question. Several solutions like STRIPS, languages $\mathcal{A}, \mathcal{B}, \mathcal{C}$ or $PDDL$, have been proposed in the literature, each with different applications in mind. A set of laws of each language is called an action program description. The semantics of each language is determined by a transition system which depends on the action program description.

Let $\langle\{\texttt{true}, \texttt{false}\}, \mathcal{F}, \mathcal{A}\rangle$ be the signature of an action language, where $\mathcal{F}$ is the set of fluent names and $\mathcal{A}$ is the set of action names in the language. Let $\langle S, V, R\rangle$ be a transition system where S is the set of all possible states, V is the evaluation function from $\mathcal{F} \times S$ into $\{\texttt{true}, \texttt{false}\}$, and finally R is the set of possible relations in the system defined as a subset of $S \times \mathcal{A} \times S$. We assume a function $\mathcal{T}(E)$ that from action program E defines the transition system $\langle S, V, R\rangle$ associated with E, and the previously defined signature. We also define $\mathcal{L}^a = \mathcal{F} \cup \mathcal{A}$.

Equipped with such notions, an $\mathcal{ETR}$ external state is a pair, with the program E describing the external domain and a state of the transition system. Then the general external oracle $\mathcal{O}^e$ is (where $\mathcal{T}(E) = \langle S, V, R\rangle$):

1. $\mathcal{O}^e((E, s), (E, s')) \models action$ iff $action \in \mathcal{A} \wedge \langle s, action, s'\rangle \in R$
2. $\mathcal{O}^e((E, s), (E, s)) \models fluent$ iff $fluent \in \mathcal{F} \wedge V(fluent, s) = \texttt{true}$

To be more concrete, let us show the instantiation of this general oracle, with Action Language $\mathcal{B}$, an action language simple enough for the purpose of this illustration, but still interesting by its ability to describe the direct and indirect effects of actions.

A program E in language $\mathcal{B}$ is composed by static laws and dynamic laws. A static law is a statement of the form: "L **if** F" where L is a literal and F a conjunction of literals. A dynamic law has the form: "A **causes** L **if** F" where A is an action name, L is a literal and F a conjunction of literals.

For the semantics of $\mathcal{B}$, the states are simply sets of literals. Then, a central notion is the concept of closure under the static laws. This says that a set of literals s is closed under a set of laws E if, for every rule "L **if** F" in E such that $F \subseteq s$, then L must belong to s. Based on this, $Cn_E(s)$ is denoted the set of *consequences* of s under E, and defined as the smallest set of literals that contains s and is closed under E. Finally, Effects_E is a function that gets the effects of a given action A in a state s based on the dynamic laws specified in E: $\text{Effects}_E(A, s) = \{L : there\ exists\ A\ \textbf{causes}\ L\ \textbf{if}\ F\ in\ E\ and\ F \subseteq s\}$.

With this, the oracle for $\mathcal{B}$ can be defined as:

1. $\mathcal{O}^e((E, s), (E, s')) \models action$ iff $s' = Cn_E(\text{Effects}_E(action, s) \cup (s \cap s'))$
2. $\mathcal{O}^e((E, s), (E, s)) \models fluent$ iff $fluent \in s$

For further detail on $\mathcal{B}$, and in particular on how the frame problem is dealt with by the above definition of s', we refer to [25].

4.2.3. Situation and Event Calculus Oracles

Action Languages are just one possible language for modeling the external environment. If one already has a domain modeled by an alternative language, such as the Situation Calculus or the Event calculus, than, in order to integrate it with $\mathcal{ETR}$, one has to define the appropriate oracle for them. This is something that can be easily done, as we show in this section, where we also illustrate its combined usage with $\mathcal{ETR}$ in some simple examples.

In the seminal Situation Calculus [42], external domains are described in a second-order language with a basic ontology partitioned into *actions* ($\mathcal{A}$), *fluents* ($\mathcal{F}$) and *situations*. An action is a predicate that has the ability to change the state of the world, while a fluent is a predicate whose truth value can change over time (or more precisely, situations). Finally, a situation represents the complete state of the universe at a given instance defined by a finite sequence of actions. More precisely, situations are either represented by a constant s_0 denoting an initial situation, or by $do(a, s)$ denoting the situation that results from executing action a in situation s.

The conditions for executing actions, and their effects, are expressed by using the second order predicates $Poss(a, s)$, meaning that action a can be executed in situation s, and $Holds(f, s)$, meaning that fluent f is true in situation s.

The semantics of these predicates and operators is defined by *axioms* describing the world, actions and their effects. For the purpose of this illustration, we do not elaborate on how these axioms are defined or how the frame problem is solved, and refer e.g. to [50] for more details. All we assume is a satisfaction relation $\models_{SitCal}$ that satisfies primitive formulas w.r.t. a set of axioms that we define as a domain description E. Intuitively, a domain description is just a set of action axioms, domain axioms and frame axioms.

Based on this, the external language is $\mathcal{L}_a = \mathcal{A} \cup \mathcal{F}$, and external states are pairs (E, S), where S is a situation and E is the external domain description. Finally, an external oracle based on Situation Calculus can be given by:

1. $\mathcal{O}^e((E,S)),(E,S)) \models f$ iff $f \in \mathcal{F}$ and $E \models_{SitCal} Holds(a,S)$
2. $\mathcal{O}^e((E,S_1),(E,S_2)) \models a$ iff $a \in \mathcal{A}$ and $E \models_{SitCal} Poss(a,S_1) \wedge S_2 = do(a,S_1)$

Note that the external oracle only executes actions that are *possible* to be executed in a given situation s_1. This precludes the system to evolve into an inconsistent situation that results from an action that is not allowed in that state. This also results in the possibility of failed external actions, which are then dealt in $\mathcal{ETR}$ by rolling back the internal KB and executing compensating actions externally.

Equipped with a formalism that is able to deal both with internal KBs, with ACID transactions, and with external actions, let us show some simple illustrative examples of what it can express, and how results are obtained.

Example 17 (Medical Diagnosis)**.** *Recall example 9. After defining the transaction rules and the internal knowledge described over a Description Logic, we left open the definition of the external KB.*

This external KB, describing the effects of actions, and also some facts about the patient, can be modeled using Situation Calculus descriptions, e.g. including:

$Holds(temperature(sam, 39), s_0).$
$Holds(hasHeadache(sam, t), s_0).$
$Holds(stuffyNose(sam, t), s_0).$
$Holds(heartRate(sam, 80), s_0).$
$Holds(dyspnea(sam, f), s_0).$
$Holds(heartRate(sam, 160), \mathtt{do}(giveMeds(sam, plp), s_0)).$
$Holds(dyspnea(sam, t), \mathtt{do}(giveMeds(sam, plp), s_0)).$

Given this instantiation of the external domain, the system will conclude that Sam likely suffers from flu and thus it may decide to give the medicine plp as treatment. If this is the case, then Sam will experience some symptoms of heart failure: dyspnea (difficulty of breathing) and increase of heart rate. Note that if this happens, it is crucial to perform some compensation in order to make Sam feel better. In this case, the system can give Sam $cplp$ that is known to address the effects of a heart failure resulting from giving plp.

Similarly, in Event Calculus [36] predicates can be *actions* or *fluents*, where actions can change properties of the world and fluents denote the properties whose truth value may change. The main feature of Event Calculus is that actions are *events*, i.e., changes associated with a particular moment in time that influence the state of the world. Then, fluents are evaluated w.r.t. time points usually defined by non-negative real numbers and denoting an explicit moment in the system.

An external domain is described in Event Calculus by the predicates[5]: $initially(f)$, denoting that fluent f hold at time 0; $initiates(f, a, t)$, stating that action a initiates fluent f at time t; $terminates(f, a, t)$ stating that action a terminates fluent f at time t; and $happens(a, t)$ denoting that action a happened at time t. Truth of fluents at time points is obtained by the predicate $holdsAt(f, t)$, whose meaning can be obtained by a logic program:

$holdsAt(P,T) \leftarrow 0 \leq T, initially(P),$
$\quad \mathbf{not}\ clipped(0,P,T).$
$holdsAt(P,T) \leftarrow happens(E_1,T_1), T_1 < T,$
$\quad initiates(E_1,P,T_1), \mathbf{not}\ clipped(T_1,P,T).$
$clipped(T_1,P,T) \leftarrow happens(E_2,T_2), T_1 < T_2, T_2 < T,$
$\quad terminates(E_2,P,T_2).$

Based simply on this, one may represent states of an external domain described in Event Calculus as pair, with a logic program P containing the description of the domain, and a time point t. The definition of the oracle itself can be done in a very similar way as in the Situation Calculus case, by:

1. $\mathcal{O}^e((P,t),(P,t)) \models p$ iff $P \vdash_{LP} holdsAt(p,t)$
2. $\mathcal{O}^e((P,t),(P',t+1)) \models a$ iff $P' = P \cup \{happens(a,t)\}$

Some words are in order regarding this representation of (external) states. Internal formulas (i.e., queries evaluated in $\mathcal{O}^d$, updates evaluated in $\mathcal{O}^t$, or complex formulas combining these) do not change the external state. Consequently, with our representation of states, from the perspective of the external domain the evaluation of all these formulas are instantaneous. In other words, this definition does not cater for cases where the external domain changes while the formulas are being evaluated. Allowing changes in the external world to occur simultaneously with the evaluation of internal formulas would require some explicit representation of the external time in the formulas of $\mathcal{ETR}$ theory, as well as a global clock with the role to instanti-

[5]We assume the simplified version of the calculus as defined in [53]. More basic predicates can be found in the full version of the calculus.

ate correctly the time component of the external state, and this is beyond the scope of this paper.

Another aspect worth discussing, is that with this formalization of the oracle, "actions" never fail. This is so because the Event Calculus was primarily defined to reason about events, and thus it makes no sense for the occurrence of an event to fail. However, $\mathcal{ETR}$, as a logic that talks about actions and state change, assumes that actions (and, especially, external actions) can fail. In fact, it is important for the internal knowledge base to know whether a given external action can be successfully executed. Without this possibility, the need to ensure transaction properties externally, as well as the notion of compensation, become redundant.

To include the possibility of failure, we extend the Event Calculus oracle, with $executable(A, T)$ to express that action A can be executed at time T. Since in the end, Event Calculus can be defined as a logic program, incorporating a new predicate is as simple as defining a new rule as $executable(A, T) \leftarrow preconditions$, where the $preconditions$ denote the set of preconditions that need to be true in order for $executable(A, T)$ succeed. For instance, in the well-known Yale shooting problem [30], one can express the possibility of killing turkey Fred as follows:

$$executable(kill, T) \leftarrow holdsAt(alive, T), \\ holdsAt(loaded, T).$$

Based on this, an alternative version of $\mathcal{O}^e$ can be defined as:

1. $\mathcal{O}^e((P, t), (P, t)) \models p$ iff $P \vdash_{LP} holdsAt(p, t)$
2. $\mathcal{O}^e((P, t), (P', t + 1)) \models a$ iff $P \vdash_{LP} executable(a, t) \wedge P' = P \cup \{happens(a, t)\}$

Example 18 (Ski Resort Hotel)**.** *Consider the scenario of a hotel in a ski resort where the internal KB manages room reservations. Given the location of the hotel, one possible package is to combine a hotel room with the acquisition of the ski pass for the resort. Moreover, the price of this package depends on the dates of the calendar (if it is high season or not) but also on the amount of snow on the slopes. If the amount of snow on the slopes is higher than 100cm then the quality is considered "premium", and the Hotel takes this opportunity to increase the price of the ski-pass reservation by 30%. However, if the slopes are closed due to some storm or lack of snow, the ski pass cannot be sold.*

Ski passes are external to the system and handled by the external environment which also gives information about the resort, namely: the quantity of snow on the slopes and if the resort is open or not.

$$\begin{aligned}
&priceFF(Price, T) \leftarrow \mathbf{ext}(\texttt{isOpen}) \otimes \mathbf{ext}(\texttt{snowCM(CM)}) \\
&\quad \otimes CM <= 100 \otimes basePrice(Price, T) \\
&priceFF(Price, T) \leftarrow \mathbf{ext}(\texttt{isOpen}) \otimes \mathbf{ext}(\texttt{snowCM(CM)}) \\
&\quad \otimes CM > 100 \otimes basePrice(P, T) \otimes Price\ is\ 1.3 * P \\
&reservation(N, T) \leftarrow priceFF(PF, T) \otimes \\
&\quad priceHotel(PH, T) \otimes addResHotel(N, T, PF + PH) \\
&\quad \otimes \mathbf{ext}(\texttt{printFF}, \texttt{cancelFF}) \\
&\quad \otimes \mathbf{ext}(askPayment(N, X)) \\
&addResHotel(N, T, X) \leftarrow roomsAvailable(Nr) > 0 \\
&\quad \otimes roomsAvailable(Nr).del \\
&\quad \otimes roomsAvailable(Nr - 1).ins \\
&\quad \otimes reservation(N, T, X).ins
\end{aligned}$$

In this case, the external domain of the ski resort could be described by an Event Calculus program with:

$$\begin{aligned}
&holdsAt(isClosed, T) \leftarrow holdsAt(stormStart, T_1), \\
&\quad T_1 \leq T, T_1 \leq T_2 \leq T, \mathbf{not}\ holdsAt(stormEnd, T_2). \\
&holdsAt(isOpen, T) \leftarrow \mathbf{not}\ holds(isClosed, T). \\
&holdsAt(stormStart, 150313). \\
&holdsAt(stormEnd, 180314). \\
&holdsAt(snowCM(10, 150313). \\
&holdsAt(snowCM(150, 183114).
\end{aligned}$$

External predicates like `isOpen` *and* `snowCM(CM)` *rely on weather conditions whose truth value naturally depend on moments in time. In the example we know that between 15th and 18th of March a snow storm occurred. During this snow storm the resort was closed and thus the hotel was unable to sell reservations with ski passes for that period. However, after this storm, the amount of snow increased and the slopes on the 18 of March had around 1,5 meters of fresh snow which led to more expensive reservations.*

Note that time is an important component of this system. It is assumed that a shared clock exists for both internal and external component. Whenever a new reservation request $reservation(name, time)$ is posed, the system must check whether the program executionally entails this transaction, given an initial internal state and external with a common appropriate value for $time$.

4.3. Combining $n-$ary External Oracles

Up until now we have considered external oracles described by a single semantics. However, example 18 illustrates a situation where more than one external semantics is required to describe the external domain. Particularly, in that example, besides the ski resort, the

hotel system interacts with one more external entity: the client. And clearly, the external action of asking the payment to a client is performed in a completely independent domain than the action `printFF`. Other examples of this need are common of the Semantic Web context where a system needs to combine knowledge published across different web-sources described over different W3C standards.

Although formally $\mathcal{ETR}$ only supports integration with one external oracle, nothing prevents this oracle to be instantiated with more than one external semantics. This can be done by partitioning the external KB language ($\mathcal{L}_a$) into as many languages as needed. Then, the oracle $\mathcal{O}^e$ works as a "meta-oracle" deciding in which semantics a formula should be evaluated. In the case of example 18, to define the two domains, viz. the ski resort and the client, then two sub-oracles (one for each domain) must be defined and incorporated within $\mathcal{O}^e$.

Assuming a disjoint language on the two sub-oracles allows $\mathcal{O}^e$ to simply decide in what semantics each formula must be evaluated. This approach can also be used to employ an arbitrary number of oracles.

5. Related Work

Although several logics exist to model transaction behavior, to the best of our knowledge, there is none with $\mathcal{ETR}$'s characteristics, where one can reason *and* execute transactions simultaneously in internal KB and external environment defined by an abstract semantics.

In this sense, logics like Action Languages [25], the Situation Calculus [42], the Event Calculus [36], Process Logic [31], PDL [19], and some of their variations like [21,40,35,37,59], provide means to reason about state change and the related phenomena of time and action. However, the goal of such logics is to describe very expressibly the dynamics of a given domain, by reasoning about the possible actions that can be executed and their (direct and indirect) effects on the domain. Thus, they provide very expressive language constructors and focus heavily in resolving the frame problem.

$\mathcal{ETR}$ (as $\mathcal{TR}$) was designed with a different intent, as its semantics centers in the combination of atomic primitives to define transactions and programs. Moreover, the meaning of these atomic primitives is abstracted from the semantics definition, and $\mathcal{ETR}$ does not provide any solution for the frame problem. For that, $\mathcal{ETR}$ theory uses three oracles as a parameter describing the dynamics of the internal and external domain. Thus, rather than an alternative to $\mathcal{ETR}$, these logics can be used together with $\mathcal{ETR}$ via the formalization of these oracles, providing instantiations of $\mathcal{ETR}$ states and meaning to its atomic primitives. Particularly, just as we present oracle instantiations for an Action Language and Situation Calculus theory, we could have presented alternative external oracle instantiations for [36,40,37,59]. For a detailed comparison between $\mathcal{TR}$ and some of these logics the interested reader is referred to [6].

Additionally, such semantics can also be used to reason about the possible compensations for a given external action. In this sense, we can lift the programmer's burden of always defining the compensating actions directly in the program, and automatically obtain the correct compensation of a given action according to a domain description. This notion was further developed by us in [28] for an action language oracle. There, we employ a relaxed notion of action reversals proposed in [15] to define an external oracle able to instantiate an external action with its correct compensation.

Moreover, since $\mathcal{ETR}$'s proof theory enables the execution of transactions, one can also compare $\mathcal{ETR}$ to formalisms like [10,58,3]. These provide tools to describe the interactions and communications between concurrent processes during long-running transactions. For that, they are based on algebraic systems for modeling concurrent communicating processes, as Milner's CCS [44] or Hoare's CSP [32], among others.

Clearly, one big difference between $\mathcal{ETR}$ and these calculus based solutions is that $\mathcal{ETR}$ does not support concurrency and synchronization. Yet, providing these features to $\mathcal{ETR}$ is an obvious future work milestone and is in line with what has been done in Concurrent Transaction Logic [7]. However, solutions like [10,58,3] are conceptually very different from $\mathcal{ETR}$. Since their focus is mostly on the correctness of conversations between processes, they provide a very powerful operational semantics to ensure correctness and termination of execution. This allows the enclosing of rich operators that for instance, can construct the correct compensation for each action "on-the-fly" as in [58]. However, these solutions are mostly operational and fail to be used as knowledge representation formalisms. Their lack of model theory and knowledge of state makes it impossible to model what is true at each step of the execution or to specify constraints on their execution based on this knowledge.

$\mathcal{ETR}$ stands in between the two worlds. It provides a clean model-theoretic semantics, parametric on the

meaning of the particular KB on which it operates, allowing us to talk about properties of transactions like equivalence and implication that hold independently of what execution path is chosen. But also, by providing a proof-theory that is sound and complete w.r.t. the semantics, $\mathcal{ETR}$ is able to talk about a particular execution of a transaction and what are the possible evolution paths for a given formula. Moreover, given its abstraction of states and primitives, $\mathcal{ETR}$ can be easily adapted for a wide range of situations, being specially useful in open contexts where several different semantics can be applicable, as e.g. in the Semantic Web.

Another interesting related work that tackles the issue of modeling transactions in arbitrary domains is the rule-based language ULTRA [17]. ULTRA is based on minimal model semantics and is very similar to $\mathcal{TR}$ (in fact, the logics are proven to have the same modeling power for their sequential version). Similarly to $\mathcal{ETR}$, ULTRA's implementation allows the definition of compensating subtransactions for every transaction committed. However, contrary to $\mathcal{ETR}$ this notion is not reflected in ULTRA's model theory which does not provide means to weaken the transactional model. Thus there is no formal correspondence between the procedure of ULTRA and its model theory as in $\mathcal{ETR}$.

In [51] the authors propose an extension of Transaction Logic with Partially Defined Actions ($\mathcal{TR}^{PAD}$) to encode axioms defining direct and indirect effects of actions and to directly define partial descriptions of states. This allows $\mathcal{TR}^{PAD}$ to model external environments with incomplete information and reason about actions and their effects in the domain. Its proof-theory, being sound and complete with the model theory allows $\mathcal{TR}^{PAD}$ to also execute these actions. However, $\mathcal{TR}^{PAD}$ deals with a different problem than $\mathcal{ETR}$. While $\mathcal{TR}^{PAD}$'s expressive power makes it more interesting to deal with external domains of which we have partially knowledge, dealing simultaneously with an internal and external environment is out of scope of its theory. Moreover, it is also impossible to relax transaction properties in $\mathcal{TR}^{PAD}$ as in $\mathcal{ETR}$.

Statelog [38] is a logic-programming like language with support for atomic transactions that has some interesting features as the ability to encode reactive rules and results about termination of programs. A fundamental difference between Statelog and $\mathcal{ETR}$ is its Kripke-style semantics based on *states* rather than *paths*. As a result, to encode evolution, states are hard-wired in the syntax each predicate as $p[S](t_1, \ldots, t_n)$ or $p(S, t_1, \ldots, t_n)$, meaning that $p(t_1, \ldots, t_n)$ holds in state S. Furthermore, although simple transactions can still be defined using rules, nested transactions need the notion of *procedures*. In these, one needs to define explicitly when a transaction must fail and commit, making nested transactions harder to encode (cf. $\mathcal{TR}$ and $\mathcal{ETR}$). Moreover, Statelog does not consider an interaction with an external entity as $\mathcal{ETR}$, and so it does not provide any mechanisms for relaxing transactional properties.

Finally, as a Semantic Web related solution we reference the RDF Triggering Language (RDFTL) [49]. RDFTL is a Event-Condition-Action language for RDF metadata on P2P environments, and also deals with the problem of interacting with external entities (i.e., other peers). Similar to $\mathcal{ETR}$, the authors agree that in such conditions it is necessary to relax atomicity and isolation properties of transactions. With this goal RDFTL also implements compensations and allows for concurrent transactions. However, RDFTL rule operational semantics as well as its definition of compensations are completely procedural and lack from a declarative semantics.

6. Conclusions

In this work we provided a complete formalization of $\mathcal{ETR}$, an extension of Transaction Logic to reason and execute (trans)actions involving updates in both an internal and external component. More precisely, with a model-theoretic semantics that has a sound and complete proof procedure, $\mathcal{ETR}$ is useful to describe transactions in systems that on the one hand have to perform updates in an internal KB, and on the other, have to perform actions externally. Examples of such systems are commonly found on the web domain, as e.g. a web-service with an internal database interacting with another (external) web-service, or a web-source with an ontology described by a Description Logic that needs to consult and execute actions in other web-sources.

In fact, the main motivation for this work arised from the Semantic Web domain. A basic requirement of the Semantic Web is the ability to reason and retrieve knowledge simultaneously from multiple web-sources described using one of the several W3C standards. Additionally, the need to reason differently according to the internal or external provenance of knowledge has been the primer motivation for the works of e.g. [33,14,29], aiming to integrate closed and open world reasoning for this Web context. Simply put, closed world reasoning assumes that everything

that is not known to be true is false. On the other hand, open world reasoning denies this principle by assuming that the current description of the world is incomplete and thus, the lack of ability to infer knowledge never implies falsity. While this latter reasoning makes sense in a open web context where one cannot presume to have complete knowledge of the environment, this is not the case when reasoning about internal knowledge. Since we fully control internal information, employing closed world assumption is useful and much more natural. To address this, several semantics like [45,47,34] have been proposed to reason and retrieve knowledge in the so called *hybrid knowledge bases*, i.e., knowledge bases described by both a non-monotonic internal KB (defined by rules) and a monotonic external Web Ontology (defined by a Description Logic).

Furthermore, the highly dynamic facet inherent to a Web environment has triggered the appearance of update operators proposals for updating and revising knowledge over Description Logics [26,41,39] but also over these hybrid knowledge bases [54,55].

If with static knowledge, it makes sense to differentiate between reasoning done with internal knowledge (where we have full control, and can apply closed world assumption) and reasoning done with knowledge coming from external sources (over which one has no control, and should thus consider open world reasoning), the same should also be done when considering the dynamics of knowledge. In fact, when acting in the internal knowledge, over which we have full control, one can always assume that any change can be undone if something fails in the execution of a transaction. On the other hand, when acting in an external environment, this can no longer be assumed. As such, a logic to deal with this complex and hybrid environment must be able to explicitly specify what to do in case of failure. In other words, the logic must be able to deal with compensations whenever there is a failure. This is exactly what $\mathcal{ETR}$ provides, and that was missing in the original $\mathcal{TR}$.

Another important feature of $\mathcal{ETR}$, inherited from $\mathcal{TR}$, is that its semantics of states and primitive operations is *abstracted*. I.e., $\mathcal{ETR}$ allows reasoning about transactions independently of the semantics chosen for the internal KB and for the external environment. This is achieved by defining oracles as a parameter of the theory, which define the most appropriate semantics for the application in question, and allows $\mathcal{ETR}$ to be useful in a wide range of domains. The flexibility obtained by the inclusion of oracles is also useful in the Semantic Web domain, in which there is a variety of possible semantics to deal with knowledge bases. Thus, by defining the appropriate oracle specification, we can apply $\mathcal{ETR}$ to the several kinds of Semantic Web knowledge bases, and also for each of the possible W3C standard usages of these knowledge bases.With this in mind, in this paper we illustrated how to build such an oracle for the *DL-Lite* semantics, the backbone of the OWL-2 QL profile.

Several semantics to describe external environments may also be useful, for instance to model interactions with external entities or agents (as in example 9). For that, languages with the goal of encoding the dynamic effects of actions in external environments are natural candidates. To achieve this usage, we have also illustrated instantiations of the external oracle for environments whose behavior can be described by Action Languages, Situation Calculus, or Event Calculus.

Because $\mathcal{ETR}$ "outsources" the decision of what is a state and what formulas hold in what states and state transitions, the goal of $\mathcal{ETR}$ is substantially different from other logics of state change. Particularly, $\mathcal{ETR}$ centers on the notion of execution paths that satisfy a given transaction, rather than what formulas hold in each state. Nevertheless, when used for reasoning, $\mathcal{ETR}$ can talk about general properties of transactions that hold in *every* possible path of execution, as e.g. saying that in every possibility of execution, transaction ϕ implies the execution of transaction ψ, or what constraints are always true in every execution of ϕ. In addition, when used for execution, $\mathcal{ETR}$ can find the particular paths where the transaction is successfully executed w.r.t. the model theory (if such a path exists).

References

[1] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.

[2] Franz Baader, Carsten Lutz, Maja Milicic, Ulrike Sattler, and Frank Wolter. Integrating Description Logics and action formalisms: First results. In *AAAI*, pages 572–577, 2005.

[3] Laura Bocchi, Cosimo Laneve, and Gianluigi Zavattaro. A calculus for long-running transactions. In Elie Najm, Uwe Nestmann, and Perdita Stevens, editors, *FMOODS*, volume 2884 of *Lecture Notes in Computer Science*, pages 124–138. Springer, 2003.

[4] Anthony J. Bonner and Michael Kifer. Transaction Logic Programming. In *ICLP'93*, pages 257–279, Cambridge, MA, USA, 1993. MIT Press.

[5] Anthony J. Bonner and Michael Kifer. Applications of Transaction Logic to knowledge representation. In *ICTL*, pages 67–81, 1994.

[6] Anthony J. Bonner and Michael Kifer. Transaction Logic Programming (or a logic of declarative and procedural knowledge). Technical Report CSRI-323, Univ. of Toronto, 1995.

[7] Anthony J. Bonner and Michael Kifer. Concurrency and communication in Transaction Logic. In *JICSLP*, pages 142–156. MIT Press, 1996.

[8] Anthony J. Bonner and Michael Kifer. Results on reasoning about updates in Transaction Logic. In *Transactions and Change in Logic Databases*, pages 166–196, 1998.

[9] Anthony J. Bonner, Michael Kifer, and Mariano P. Consens. Database programming in Transaction Logic. In Catriel Beeri, Atsushi Ohori, and Dennis Shasha, editors, *DBPL*, Workshops in Computing, pages 309–337. Springer, 1993.

[10] Michael J. Butler, C. A. R. Hoare, and Carla Ferreira. A trace semantics for long-running transactions. In Ali E. Abdallah, Cliff B. Jones, and Jeff W. Sanders, editors, *25 Years Communicating Sequential Processes*, volume 3525 of *Lecture Notes in Computer Science*, pages 133–150. Springer, 2004.

[11] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in Description Logics: The DL-Lite family. *Journal of Automated reasoning*, 39(3):385–429, 2007.

[12] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. DL-Lite: Tractable Description Logics for ontologies. In *AAAI*, pages 602–607, 2005.

[13] Diego Calvanese, Evgeny Kharlamov, Werner Nutt, and Dmitriy Zheleznyakov. Updating ABoxes in DL-Lite. In Alberto H. F. Laender and Laks V. S. Lakshmanan, editors, *AMW*, volume 619 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2010.

[14] Carlos Viegas Damásio, Anastasia Analyti, Grigoris Antoniou, and Gerd Wagner. Supporting Open and Closed World Reasoning on the Web. In José Júlio Alferes, James Bailey, Wolfgang May, and Uta Schwertel, editors, *PPSWR*, volume 4187 of *Lecture Notes in Computer Science*, pages 149–163. Springer, 2006.

[15] Thomas Eiter, Esra Erdem, and Wolfgang Faber. On reversing actions: Algorithms and complexity. In *IJCAI*, pages 336–341, 2007.

[16] Thomas Eiter, Esra Erdem, Michael Fink, and Ján Senko. Updating action domain descriptions. *Artif. Intell.*, 174(15):1172–1221, 2010.

[17] Alfred Fent, Carl-Alexander Wichert, and Burkhard Freitag. Logical update queries as open nested transactions. In Gunter Saake, Kerstin Schwarz, and Can Türker, editors, *FMLDO - Selected Papers*, volume 1773 of *Lecture Notes in Computer Science*, pages 45–66. Springer, 1999.

[18] Richard Fikes, Patrick Hayes, and Ian Horrocks. OWL-QL: a language for deductive query answering on the Semantic Web. *Web semantics: Science, services and agents on the World Wide Web*, 2(1):19–29, 2004.

[19] Michael J. Fischer and Richard E. Ladner. Propositional dynamic logic of regular programs. *J. Comput. Syst. Sci.*, 18(2):194–211, 1979.

[20] Paul Fodor and Michael Kifer. Transaction Logic with defaults and argumentation theories. In *ICLP (Technical Communications)*, pages 162–174, 2011.

[21] Alfredo Gabaldon. Non-markovian control in the Situation Calculus. In Rina Dechter and Richard S. Sutton, editors, *AAAI/IAAI*, pages 519–525. AAAI Press / The MIT Press, 2002.

[22] Hector Garcia-Molina and Kenneth Salem. Sagas. *SIGMOD Rec.*, 16:249–259, Dec. 1987.

[23] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general Logic Programs. *J. ACM*, 38(3):620–650, 1991.

[24] Michael Gelfond and Vladimir Lifschitz. The stable model semantics for Logic Programming. In *ICLP/SLP*, pages 1070–1080, 1988.

[25] Michael Gelfond and Vladimir Lifschitz. Action Languages. *Electron. Trans. Artif. Intell.*, 2:193–210, 1998.

[26] Giuseppe De Giacomo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. On instance-level update and erasure in Description Logic ontologies. *J. Log. Comput.*, 19(5):745–770, 2009.

[27] Ana Sofia Gomes and José Júlio Alferes. Extending Transaction Logic with external actions (technical communication). *Theory and Practice of Logic Programming, On-line Supplement*, 2013.

[28] Ana Sofia Gomes and José Júlio Alferes. External Transaction Logic with automatic compensations. In João Leite, Tran Cao Son, Paolo Torroni, Leon van der Torre, and Stefan Woltran, editors, *CLIMA*, volume 8143 of *Lecture Notes in Computer Science*, pages 239–255. Springer, 2013.

[29] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description Logic Programs: combining Logic Programs with Description Logic. In Gusztáv Hencsey, Bebo White, Yih-Farn Robin Chen, László Kovács, and Steve Lawrence, editors, *WWW*, pages 48–57. ACM, 2003.

[30] S. Hanks and D. McDermott. Nonmonotonic logic and temporal projection. *Artif. Intell.*, 33(3):379–412, November 1987.

[31] David Harel, Dexter Kozen, and Rohit Parikh. Process Logic: Expressiveness, decidability, completeness. *J. Comput. Syst. Sci.*, 25(2):144–170, 1982.

[32] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[33] Ian Horrocks, Peter F Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, Mike Dean, et al. SWRL: A Semantic Web rule language combining OWL and RuleML. *W3C Member submission*, 21:79, 2004.

[34] Matthias Knorr, José Júlio Alferes, and Pascal Hitzler. Local closed world reasoning with description logics under the well-founded semantics. *Artif. Intell.*, 175(9-10):1528–1554, 2011.

[35] Robert A. Kowalski. Database updates in the Event Calculus. *J. Log. Program.*, 12(1&2):121–146, 1992.

[36] Robert A. Kowalski and Marek J. Sergot. A logic-based Calculus of Events. *New Generation Comput.*, 4(1):67–95, 1986.

[37] Kim Guldstrand Larsen and Bent Thomsen. A Modal Process Logic. In *LICS*, pages 203–210. IEEE Computer Society, 1988.

[38] Georg Lausen, Bertram Ludäscher, and Wolfgang May. On active deductive databases: The Statelog approach. In *Transactions and Change in Logic Databases*, pages 69–106, 1998.

[39] Maurizio Lenzerini and Domenico Fabio Savo. Updating inconsistent Description Logic knowledge bases. In Luc De Raedt, Christian Bessière, Didier Dubois, Patrick Doherty, Paolo Frasconi, Fredrik Heintz, and Peter J. F. Lucas, editors, *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 516–521. IOS Press, 2012.

[40] Hector J. Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin, and Richard B. Scherl. GOLOG: A Logic Pro-

gramming language for dynamic domains. *J. Log. Program.*, 31(1-3):59–83, 1997.

[41] Hongkai Liu, Carsten Lutz, Maja Milicic, and Frank Wolter. Foundations of instance level updates in expressive Description Logics. *Artif. Intell.*, 175(18):2170–2197, 2011.

[42] John McCarthy. Situations, actions, and causal laws. Technical report, Stanford University, 1963. Reprinted in MIT Press, Cambridge, Mass., 1968 pages 410-417.

[43] Deborah L McGuinness, Frank Van Harmelen, et al. OWL web ontology language overview. *W3C recommendation*, 10(2004-03):10, 2004.

[44] Robin Milner. Calculi for synchrony and asynchrony. *Theor. Comput. Sci.*, 25:267–310, 1983.

[45] Boris Motik, Ian Horrocks, Riccardo Rosati, and Ulrike Sattler. Can OWL and Logic Programming live together happily ever after? In Isabel F. Cruz, Stefan Decker, Dean Allemang, Chris Preist, Daniel Schwabe, Peter Mika, Michael Uschold, and Lora Aroyo, editors, *International Semantic Web Conference*, volume 4273 of *Lecture Notes in Computer Science*, pages 501–514. Springer, 2006.

[46] Boris Motik, Peter F Patel-Schneider, Bijan Parsia, Conrad Bock, Achille Fokoue, Peter Haase, Rinke Hoekstra, Ian Horrocks, Alan Ruttenberg, Uli Sattler, et al. OWL 2 web ontology language: Structural specification and functional-style syntax. *W3C recommendation*, 27:17, 2009.

[47] Boris Motik and Riccardo Rosati. A faithful integration of Description Logics with Logic Programming. In *IJCAI*, pages 477–482, 2007.

[48] Boris Motik, Ulrike Sattler, and Rudi Studer. Query answering for OWL-DL with rules. *J. Web Sem.*, 3(1):41–60, 2005.

[49] George Papamarkos, Alexandra Poulovassilis, and Peter T. Wood. Event-condition-action rules on RDF metadata in P2P environments. *Computer Networks*, 50(10):1513–1532, 2006.

[50] Raymond Reiter. The frame problem in the Situation Calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *AI and mathematical theory of computation*, pages 359–380. Academic Press, San Diego, CA, USA, 1991.

[51] Martín Rezk and Michael Kifer. Transaction Logic with partially defined actions. *J. Data Semantics*, 1(2):99–131, 2012.

[52] Dumitru Roman and Michael Kifer. Reasoning about the behavior of Semantic Web Services with Concurrent Transaction Logic. In *VLDB*, pages 627–638, 2007.

[53] Murray Shanahan. The Event Calculus explained. In *AI Today*, pages 409–430. 1999.

[54] Martin Slota and João Leite. Towards closed world reasoning in dynamic open worlds. *TPLP*, 10(4-6):547–563, 2010.

[55] Martin Slota, João Leite, and Terrance Swift. Splitting and updating hybrid knowledge bases. *TPLP*, 11(4-5):801–819, 2011.

[56] Michael Thielscher. Introduction to the Fluent Calculus. *Electron. Trans. Artif. Intell.*, 2:179–192, 1998.

[57] Ivan José Varzinczak. On action theory change. *J. Artif. Intell. Res. (JAIR)*, 37:189–246, 2010.

[58] Cátia Vaz and Carla Ferreira. On the analysis of compensation correctness. *J. Log. Algebr. Program.*, 81(5):585–605, 2012.

[59] Dongmo Zhang and Norman Y. Foo. Epdl: A logic for causal reasoning. In Bernhard Nebel, editor, *IJCAI*, pages 131–138. Morgan Kaufmann, 2001.

Appendix

In this appendix we present the proofs of the results in the paper. Some of these proofs require auxiliary results and definitions, that are present later, in Section B. To help the reader, the following table details the dependences between the auxiliary and the main results.

Main Result	*Auxiliary Results Needed*
Proposition 1	–
Theorem 2	–
Theorem 3	Theorem 2 and Lemmas 1-2
Theorem 4	Lemmas 5-16

A. Proofs of the main results

Proof of Proposition 1. We prove each item separately.

1. If $M, \pi \models_p \phi$ and $M, \pi \not\models_c \phi$ then $\exists a$ s.t. a occurs in ϕ, $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$

   We prove by induction on the structure of ϕ:
   Base Case: If ϕ is an atom then, by the Base Case of the definitions of Classical and Partial satisfaction (Case 1. of Definitions 13 and 14) we know that $\phi \in \mathcal{L}_i \vee \mathcal{L}_a^*$ and for a path π s.t. π is a 1-path: $M, \pi \models_c \phi$ and $M, \pi \not\models_c \phi$. Since ϕ is an action, and π is a 1-path, this statement holds for action $a = \phi$.
   Induction Step:
   Conjunction: Assume that the result holds for ϕ and ψ. We need to prove that if $M, \pi \models_p \phi \wedge \psi$ and $M, \pi \not\models_c \phi \wedge \psi$ then $\exists a$ s.t. a occurs in $\phi \wedge \psi$ $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$.
   Simplifying, we have to prove that if $M, \pi \models_p \phi$ and $M, \pi \models_p \psi$, and $M, \pi \not\models_c \phi$ or $M, \pi \not\models_c \psi$, then $\exists a$ s.t. a occurs in $\phi \wedge \psi$, $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$. We have two possible cases:
   1) Suppose that $M, \pi \models_p \phi$ and $M, \pi \models_p \psi$ and $M, \pi \not\models_c \phi$ hold. Then we can apply apply the induction hypothesis to ϕ and thus conclude that $\exists a$ s.t. a occurs in ϕ $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$. Since a occurs in ϕ, then it also occurs in $\phi \wedge \psi$, and thus the hypothesis holds in this case.
   2) Suppose that $M, \pi \models_p \phi$ and $M, \pi \models_p \psi$ and $M, \pi \not\models_c \psi$ hold. Similarly to the previous case, we can apply the induction hypothesis to ψ and conclude that $\exists a$ s.t. a occurs in ψ, $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$. Since a occurs in ψ, then it also occurs in $\phi \wedge \psi$, and thus the hypothesis holds in this case.
   Disjunction: Assume that the result holds for ϕ and ψ. We need to prove that if $M, \pi \models_p \phi \vee \psi$ and $M, \pi \not\models_c \phi \vee \psi$ then $\exists a$ s.t. a occurs in $\phi \vee \psi$, $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$.
   Simplifying, we have to prove that if $M, \pi \models_p \phi$ or $M, \pi \models_p \psi$, and $M, \pi \not\models_c \phi$ and $M, \pi \not\models_c \psi$ then $\exists a$ s.t. a occurs in $\phi \vee \psi$, $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$. Now we have two cases:
   1) If $M, \pi \models_p \phi$ and $M, \pi \not\models_p \psi$ and $M, \pi \not\models_c \phi$ hold, then we can apply the induction hypothesis to ϕ and thus conclude that $\exists a$ s.t. a occurs in ϕ, $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$. Since a occurs in ϕ, then it also occurs in $\phi \vee \psi$, and thus the hypothesis holds in this case.
   2) If $M, \pi \models_p \psi$ and $M, \pi \not\models_p \psi$ and $M, \pi \not\models_c \phi$ hold, then we can apply the induction hypothesis to ψ and thus conclude that $\exists a$ s.t. a occurs in ψ. $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$. Since a occurs in ψ, then it also occurs in $\phi \vee \psi$, and thus the hypothesis holds in this case.
   Serial Conjunction: Assume the result holds for ϕ and ψ. We need to prove that if $M, \pi \models_p \phi \otimes \psi$ and $M, \pi \not\models_c \phi \otimes \psi$ then $\exists a$ s.t. a occurs in $\phi \otimes \psi$, $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$.
   By definition of serial conjunction in $\models_p$ and $\models_c$, this is equivalent to ([(a) $\exists \pi_1, \pi_2 : \pi_1 \circ \pi_2 = \pi$ and $M, \pi_1 \models_c \phi$ and $M, \pi_2 \models_p \psi$] or [(b) $M, \pi \models_p \phi$ and $M, \pi \not\models_c \phi$]) And $\forall \pi_3, \pi_4 : \pi_3 \circ \pi_4 = \pi, M, \pi_3 \not\models_c \phi$ or $M, \pi_4 \not\models_c \psi$
   Suppose (b) is the case. We can apply the Induction Hypothesis for ϕ, and conclude that $\exists a$ s.t. a occurs in ϕ, $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$. Since a occurs in ϕ, then a occurs in $\phi \otimes \psi$, and thus the hypothesis holds in this case.
   Suppose (a) is the case and $\forall \pi_3, \pi_4 : \pi_3 \circ \pi_4 = \pi$ either $M, \pi_3 \not\models_c \phi$ or $M, \pi_4 \not\models_c \psi$ (because $M, \pi \not\models_c \phi \otimes \psi$). Since this latter statement holds for every split of π, then in particular it also holds for $\pi = \pi_1 \circ \pi_2$. As such, we know that $M, \pi_1 \models_c \phi$ and $M, \pi_2 \models_p \psi$ and ($M, \pi_1 \not\models_c \phi$ or $M, \pi_2 \not\models_c \psi$).
   Since $M, \pi_1 \models_c \phi$ and $M, \pi_1 \not\models_c \phi$ cannot be true together, we can conclude that $M, \pi_1 \models_c \phi$, $M, \pi_2 \models_p \psi$ and $M, \pi_2 \not\models_c \psi$. From this, we can apply the induction hypothesis to ψ for the path π_2 and conclude that $\exists a$ s.t. a occurs in ψ. $M, \pi'_{\texttt{end}} \models_p a$ and $M, \pi'_{\texttt{end}} \not\models_c a$ for path π_2. Since π is composed of $\pi_1 \circ \pi_2$, π_2 is in the *ending* of path π then this result holds for path π and

$\pi'_{\texttt{end}} = \pi_{\texttt{end}}$. Since a occurs in ψ, then it also occurs in $\phi \otimes \psi$ and thus the hypothesis holds for this case.
Negation: Let's assume that the result holds for ϕ. We need to prove that if $M, \pi \models_p \neg\phi$ and $M, \pi \not\models_c \neg\phi$ then $\exists a$ s.t. a occurs in $\neg\phi$, $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$.
By definition of negation in the partial and classical satisfactions, we know that this is equivalent to $M, \pi \not\models_p \phi$ and $M, \pi \models_c \phi$. However, this latter is always false as $M, \pi \models_c \phi$ implies $M, \pi \models_p \phi$ by definition of $\models_p$. Since the antecedent of the implication is always false, the statement is trivially satisfied. □

2. If $M, \pi \models_p \phi$ and $M, \pi \not\models_c \phi$ then $M, \pi \models_p \phi \otimes \psi$
This result holds immediately a consequence of item 5.a) of the definition of partial satisfaction (Definition 14).
3. Let ϕ be a positive formula: If $M, \pi \models_c \phi$ then $M, \pi \models_p \phi$
We prove this result by induction on the structure of ϕ:
Base Case: If ϕ is an atom, then this statement holds immediately by item 1.a) of the definition of partial satisfaction (Definition 14).
Induction Step:
Conjunction: Let's assume that the result holds for ϕ and ψ. We want to prove that if $M, \pi \models_c \phi \wedge \psi$ then $M, \pi \models_p \phi \wedge \psi$. As such, assume that $M, \pi \models_c \phi \wedge \psi$, and thus, by definition of classical satisfaction, we have $M, \pi \models_c \phi$ and $M, \pi \models_c \psi$. Moreover, applying the induction hypothesis, we know that $M, \pi \models_p \phi$ and $M, \pi \models_p \psi$ and thus by the classical conjunction case of definition 14 we have that $M, \pi \models_p \phi \wedge \psi$
Disjunction: Let's assume that the result holds for ϕ and ψ. We want to prove that if $M, \pi \models_c \phi \vee \psi$ then $M, \pi \models_p \phi \vee \psi$. Assuming $M, \pi \models_c \phi \vee \psi$ then we know that either $M, \pi \models_c \phi$ or $M, \pi \models_c \psi$ holds. Consequently, we have two cases:
1) $M, \pi \models_c \phi$ and since the result holds for ϕ then we know that $M, \pi \models_p \phi$ and thus by definition of the disjunction case $M, \pi \models_p \phi \vee \psi$ holds for any transaction ψ. 2) $M, \pi \models_c \psi$ and since the result holds for ψ then we know that $M, \pi \models_p \psi$ and thus by definition of the disjunction case $M, \pi \models_p \phi \vee \psi$ holds for any transaction ϕ.
Serial Conjunction: Assume that the result holds for ϕ and ψ. We want to prove that if $M, \pi \models_c \phi \otimes \psi$ then $M, \pi \models_p \phi \otimes \psi$. Since we know that $M, \pi \models_c \phi \otimes \psi$ then there must exist a split $\pi_1 \circ \pi_2$ of π s.t. $M, \pi_1 \models_c \phi$ and $M, \pi_2 \models_c \psi$. If this is the case, then by hypothesis we know that $M, \pi_1 \models_p \phi$ and $M, \pi_2 \models_p \psi$. By applying item b) of the serial conjunction case of partial satisfaction, it follows that $M, \pi \models_p \phi_1 \otimes \phi_2$.
4. $M, \pi \models_c \phi_P$ iff $M, \pi \models_p \phi_P$
This statement follow easily from the definition of partial satisfaction. Since ϕ_P is an atom from $\mathcal{L}_P$, then item 2. of Definition 14 is never applicable. As such, if $M, \pi \models_p \phi_P$ then $M, \pi \models_c \phi_P$. The reverse holds by item 1. of Definition 14

□

Proof of Theorem 2. We prove each item separately,

- If $M, \pi \models_c \phi$ then $M, \pi \models \phi$
This result is proven by induction on the structure of ϕ:
Base Case: If ϕ is an atom, then this statement holds trivially by item 1 of definitions 18 and 13.
Induction Step:
Disjunction: Assume that the result holds for ϕ and ψ. We want to prove that if $M, \pi \models_c \phi \vee \psi$ then $M, \pi \models \phi \vee \psi$. Since $M, \pi \models_c \phi \vee \psi$ then one of the following holds: (1) $M, \pi \models_c \phi$ or (2) $M, \pi \models_c \phi \vee \psi$. If (1) is the case then, by hypothesis, we know that $M, \pi \models \phi$, and thus by definition $M, \pi \models \phi \vee \psi$. If (2) is the case instead, then, by hypothesis, it follows that $M, \pi \models \psi$, and so by definition $M, \pi \models \phi \vee \psi$.
Conjunction: Let's assume that this result holds for ϕ and ψ. We want to prove that if $M, \pi \models_c \phi \wedge \psi$ then $M, \pi \models \phi \wedge \psi$. Since $M, \pi \models_c \phi \wedge \psi$ then both $M, \pi \models_c \phi$ and $M, \pi \models_c \psi$. So, by induction hypothesis, $M, \pi \models \phi$ and $M, \pi \models \psi$ holds, and thus, by definition, $M, \pi \models \phi \wedge \psi$.
Serial Conjunction: Assume now that the result holds for ϕ and ψ. We want to prove that if $M, \pi \models_c \phi \otimes \psi$ then $M, \pi \models \phi \otimes \psi$. Since $M, \pi \models_c \phi \otimes \psi$ holds, then there must exist a split $\pi_1 \circ \pi_2$ of π s.t. $M, \pi_1 \models_c \phi$ and $M, \pi_2 \models_c \psi$. Then, by induction hypothesis, we can conclude $M, \pi_1 \models \phi$ and $M, \pi_2 \models \psi$. Since $\pi_1 \circ \pi_2$ are splits of π, then by the serial conjunction case of the definition of general satisfaction (Definition 18) if follows that $M, \pi \models \phi \otimes \psi$.

- For the second item of this Theorem, we have to prove that $M, \pi \models_c \phi$ iff $M, \pi \models \phi$, where π is a path without external actions in the annotated

transitions.

Since π does not contain annotations for external actions, we know that any subpath of π_1 of π, π_1 does not have external actions. Moreover, we know by Definiton 17 that if a path π_1 does not contain external actions in the annotations, then $M, \pi_1 \rightsquigarrow \phi$ is impossible for any formula ϕ. Consequently, since $M, \pi_1 \rightsquigarrow \phi$ is impossible for any subpath π_1 of π, then the Compensating Case of definition 18 is never applicable for path π. Thus, since the definitions of general and partial satisfaction (Definitions 18 and 13) are exactly the same except for this case, then for such a path π, it follows that $M, \pi \models \phi$ iff $M, \pi' \models \phi$.

□

Proof of Theorem 3.

$$P, \langle (D_0, E) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi \text{ iff } P, \langle D_1, \ldots, D_n\rangle \models_{TR} \phi$$

We start by showing that $P, ((D_0, E) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_n}\!\!\rightarrow (D_n, E)) \models_{ETR} \phi$ implies $P, (D_1, \ldots, D_n) \models_{TR} \phi$ ($\Rightarrow$ direction), and then show the converse, i.e., that $P, (D_1, \ldots, D_n) \models_{TR} \phi$ implies $P, ((D_0, E) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_n}\!\!\rightarrow (D_n, E)) \models_{ETR} \phi$ (the $\Leftarrow$ direction). For this proof, we take into account the result of Theorem 2, which says that for paths π without external actions, $M, \pi \models_c \phi$ iff $M, \pi \models \phi$ in $\mathcal{ETR}$.

$\Rightarrow$: This proof follows by contradiction.

First recall that if $P, ((D_0, E) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_n}\!\!\rightarrow (D_n, E)) \models_{ETR} \phi$ holds, then, by definition of the executional entailment, for every $\mathcal{ETR}$ model M of program P, it is the case that $M, \langle (D_0, E) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi$.

Let's assume that $P, (D_1, \ldots, D_n) \not\models_{TR} \phi$. By definition this implies that there is a $\mathcal{TR}$ model of P, M_1, s.t. $M_1, \langle D_1, \ldots, D_n\rangle \not\models \phi$. Additionally, we can construct an $\mathcal{ETR}$ interpretation M_1^{ETR}, as defined in the auxiliary Definition 27 below, that, as proven by auxiliary Lemma 1, is an $\mathcal{ETR}$ model of P. This Lemma 1 also tells us that $M_1, \langle D_1, \ldots, D_n\rangle \models_{TR} \phi$ iff $M_1^{ETR}, \langle (D_0, E) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi$, and thus $M_1, \langle D_1, \ldots, D_n\rangle \not\models \phi$ implies that $M_1^{ETR}, \langle (D_0, E) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_n}\!\!\rightarrow (D_n, E)\rangle \not\models_{ETR} \phi$. Since M_1^{ETR} is also an $\mathcal{ETR}$ model, this leads to a contradiction, and thus $P, (D_1, \ldots, D_n) \models_{TR} \phi$ must hold.

$\Leftarrow$: Once again, we make the proof by contradiction.

Start by recalling that If $P, (D_1, \ldots, D_n) \models_{TR} \phi$, by definition of the executional entailment in $\mathcal{TR}$, it follows that for every $\mathcal{TR}$ model M of P: $M, \langle D_1, \ldots, D_n\rangle \models_{TR} \phi$.

By contradiction, assume that $P, ((D_0, E) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_n}\!\!\rightarrow (D_n, E)) \not\models_{ETR} \phi$. By definition this implies that there must exist a M_1, s.t. M_1 is an $\mathcal{ETR}$ model of P and $M_1, \langle (D_0, E) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_n}\!\!\rightarrow (D_n, E)\rangle \not\models_{ETR} \phi$. If this is the case, then we can construct a $\mathcal{TR}$ interpretation M_1^{TR}, as defined below (auxiliary Definition 28) that, c.f. auxiliary Lemma 2, is a $\mathcal{TR}$ model of P. Moreover, this Lemma 2 also tells us that $M_1^{TR}, \langle D_1, \ldots, D_n\rangle \models_{TR} \phi$ iff $M_1, \langle (D_0, E) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi$, and thus $M_1, \langle (D_0, E) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_n}\!\!\rightarrow (D_n, E)\rangle \not\models_{ETR} \phi$, implies that $M_1^{TR}, \langle D_1, \ldots, D_n\rangle \not\models_{TR} \phi$. Since M_1^{TR} is also a $\mathcal{TR}$ model, this leads to a contradiction, and $P, ((D_0, E) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_n}\!\!\rightarrow (D_n, E)) \models_{ETR} \phi$ must hold. □

Proof of Theorem 4.

$$P, \pi \models \phi \text{ iff } P, \pi \vdash \phi$$

Soundness

Soundness of Axiom:

Assume $P, \pi \vdash ()$. Then, by definition of $\vdash$, π must be a 1-path. Since we defined () as an empty transaction that holds for all paths of size 1, then $M, \pi \models ()$ for all interpretations M, and so $P, \pi \models ()$.

Soundness of Rules: This proof follow by separately proving the soundness of each of the rules in the definition of $SLD_{\mathcal{ETR}}$-derivation (rules r1–r5 of Figure 1)) as detailed in the auxiliary Lemma 9.

Completeness

This proof is inspired by the completeness proof of $\mathcal{TR}$'s Proof Theory [6]. Similarly to the proof for $\mathcal{TR}$, we construct a canonical interpretation (auxiliary Definition 31) of the program P, M_P, that intuitively collects all the results obtained by the proof theory. We then prove that M_P is a model of P (auxiliary Lemma 16).

Saying that $P, \pi \models \phi$ is equivalent to saying that for every model M of P, $M, \pi \models \phi$. Given that, as we've established, M_P is a model of P, it follows that $M_P, \pi \models \phi$.

Given that, all that remains to be proven is that if M_P satisfies a formula ϕ in a given path π, then the procedure proves ϕ in that path, i.e. $P, \pi \vdash \phi$. This is done in auxiliary Lemmas 13 and 14. □

B. Auxiliary Results

B.1. Auxiliary Results for Theorem 3

Definition 26 (Path without external actions)**.** *Let π be a path. We say that π does not contain external actions if for every annotated action A_i that appears in π, A_i is **not** an external action.*

Definition 27. *Let P be a program without external actions, well-formed in both $\mathcal{TR}$ and $\mathcal{ETR}$. Let M be a $\mathcal{TR}$ model of P. We define M^{ETR} to be the interpretation obtained from M as follows*

1. *If $\phi \in \mathcal{L}_P$ then, whenever $\phi \in M(\langle(D_0, \ldots, D_n)\rangle)$ then $\phi \in M^{ETR}(\langle(D_0, E)\ ^{A_1} \rightarrow \ldots\ ^{A_n} \rightarrow (D_n, E)\rangle)$ for every external state E and every A_i $(1 \leq i \leq n)$ s.t. $\mathcal{O}^t(D_{i-1}, D_i) \models A_i$;*
2. *If $\phi \in \mathcal{L}_\mathcal{O}$ then, for every π satisfying the restrictions in Definition 11, ϕ belongs to $M^{ETR}(\pi)$;*
3. *nothing else belongs to $M^{ETR}(\pi)$.*

Lemma 1. *Let P be a program without external actions, well-formed in both $\mathcal{TR}$ and $\mathcal{ETR}$ and ϕ be a formula without external actions, well-formed in both $\mathcal{TR}$ and $\mathcal{ETR}$. Let M be a $\mathcal{TR}$ model of P and M^{ETR} the $\mathcal{ETR}$ interpretation obtained from M as described in Definition 27. Then:*

- *M^{ETR} is a $\mathcal{ETR}$ model of P,* and
- *$M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_{ETR} \phi$ iff $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi$*

Proof. We start by proving the equivalence in the second item, and do this by proving each of the implications separately. First we show ($\Rightarrow$ direction) that $M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_{ETR} \phi$ implies $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi$.
$\Rightarrow$:
We make this proof by induction on the structure of ϕ:

- *Base Case:* ϕ is an atom, and thus by definition there are two possible scenarios:
  - $\phi \in \mathcal{L}_i$.
    Since ϕ is a primitive of the oracle we know that $M^{ETR}, \pi \models_{ETR} \phi$ if one of the two cases holds:
    1. $\pi = \langle(D, E)\rangle$ and $\mathcal{O}^d(D) \models \phi$. In this case, by definition of $\mathcal{TR}$ interpretation, we know that $\phi \in M(\langle D\rangle)$ and thus that $M, \langle D\rangle \models_{TR} \phi$
    2. $\pi = \langle(D_1, E)^{\phi}\rightarrow (D_2, E)\rangle$ and $\mathcal{O}^t(D_1, D_2) \models \phi$. By definition of $\mathcal{TR}$ interpretation, we know that $\phi \in M(\langle D_1, D_2\rangle)$ and thus that $M, \langle D_1, D_2\rangle \models_{TR} \phi$
  - $\phi \in \mathcal{L}_P$.
    $M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_{ETR} \phi$ and since this path does not have external actions, we can apply the results of Theorem 2 and $M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_{ETR} \phi$ iff $M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_c \phi$. Since $\phi \in \mathcal{L}_P$, this is equivalent to say that $\phi \in M^{ETR}(\langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle)$. Then, by definition, $\phi \in M^{ETR}(\langle(D_0, E)\ ^{A_1} \rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle)$ because $\phi \in M(\langle D_0, \ldots, D_n\rangle)$. Thus, $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi$.
- *Conjunction:* $\phi = \phi_1 \wedge \phi_2$
  $M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_{ETR} \phi_1 \wedge \phi_2$, and since this path does not have external actions in its transitions, we know from Theorem 2 that $M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_c \phi_1 \wedge \phi_2$. Moreover, this is equivalent to say that $M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_{ETR} \phi_1$ and $M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_{ETR} \phi_2$. From this, we can apply our Induction Hypothesis individually to ϕ_1 and ϕ_2, concluding $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1$ and $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_2$. Finally, by definition of $\models_{TR}$ we know that in this case $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1 \wedge \phi_2$
- *Disjunction:* $\phi = \phi_1 \vee \phi_2$
  $M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_{ETR} \phi_1 \vee \phi_2$, and since this path does not have external actions in its transitions, we know from Theorem 2 that $M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_c \phi_1 \vee \phi_2$. Moreover, this is equivalent to say that $M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_{ETR} \phi_1$ or $M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_{ETR} \phi_2$. We can then apply our Induction Hypothesis to ϕ_1 and ϕ_2 individually, concluding that $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1$ or $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_2$. By definition of $\models_{TR}$ we can conclude that $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1 \vee \phi_2$.
- *Serial Conjunction:* $\phi = \phi_1 \otimes \phi_2$
  $M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_{ETR} \phi_1 \otimes \phi_2$, and since this path does not have external actions in its transitions, we know from Theorem 2 that $M^{ETR}, \langle(D_0, E)\ ^{A_1}\rightarrow \ldots\ ^{A_n}\rightarrow (D_n, E)\rangle \models_c \phi_1 \otimes \phi_2$. Moreover, this is equivalent to saying that there is an i, where $0 \leq$

$i \leq n$, such that $M^{ETR}, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_i, E)\rangle \models_{ETR} \phi_1$ and $M^{ETR}, \langle (D_i, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle \models_{ETR} \phi_2$. We can then apply our Induction Hypothesis to ϕ_1 and ϕ_2 individually, concluding that $M, \langle D_0, \ldots, D_i\rangle \models_{TR} \phi_1$ and $M, \langle D_i, \ldots, D_n\rangle \models_{TR} \phi_2$. By definition of $\models_{TR}$ we know that this is equivalent to $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1 \otimes \phi_2$.

- *Negation:* $\phi = \neg\phi_1$
  $M^{ETR}, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle \models_{ETR} \neg\phi$, since this path does not have external actions in its transitions, from Theorem 2 we know that $M^{ETR}, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle \models_c \neg\phi_1$. Moreover, since ϕ_1 is an atom, this is equivalent to say $\phi_1 \notin M^{ETR}(\langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle)$ Then, by definition 27 we know that $\phi_1 \notin M(\langle D_0, \ldots, D_n\rangle)$ (as otherwise it would lead to a contradiction). Since ϕ_1 is an atom, $M, \pi \models_{TR} \phi_1$ iff $\phi_1 \in M(\pi)$. Thus, it is not the case that $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1$ which implies $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \neg\phi_1$.

$\Leftarrow$:
Here we show that: $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi$ implies $M^{ETR}, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle \models_{ETR} \phi$.
This is also proven by induction on the structure of ϕ:

- *Base Case:* ϕ is an atom.
  By satisfaction in $\mathcal{TR}$, $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi$ iff $\phi \in M(\langle D_0, \ldots, D_n\rangle)$. If the latter is true, by Definition 27, it follows that $\phi \in M^{ETR}(\langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle)$ which, by definition of $\models_{ETR}$, leads to $M^{ETR}, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle \models_{ETR} \phi$
- *Conjunction:* $\phi = \phi_1 \wedge \phi_2$
  From satisfaction in $\mathcal{TR}$, $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1 \wedge \phi_2$ implies that $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1$ and $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_2$. Applying the Induction Hypothesis individually to ϕ_1 and ϕ_2, it follows that $M^{ETR}, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle \models_{ETR} \phi_1$ and $M^{ETR}, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle \models_{ETR} \phi_2$. From this we can conclude as intended: $M^{ETR}, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle \models_{ETR} \phi_1 \wedge \phi_2$
- *Disjunction:* $\phi = \phi_1 \vee \phi_2$ This is proven exactly as for conjunction, replacing $\wedge$ by $\vee$.
- *Serial Conjunction:* $\phi = \phi_1 \otimes \phi_2$
  Given $\mathcal{TR}$'s satisfaction relation as presented in Definition 3: $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1 \otimes \phi_2$ implies that there is an i (where $0 \leq i \leq n$) such that $M, \langle D_0, \ldots, D_i\rangle \models_{TR} \phi_1$ and $M, \langle D_i, \ldots, D_n\rangle \models_{TR} \phi_2$. By applying the Induction Hypothesis individually to ϕ_1 and ϕ_2 we obtain that $M^{ETR}, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_i, E)\rangle \models_{ETR} \phi_1$ and $M^{ETR}, \langle (D_i, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle \models_{ETR} \phi_2$. From this, it follows that, as desired, $M^{ETR}, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle \models_{ETR} \phi_1 \otimes \phi_2$
- *Negation:* $\phi = \neg\phi_1$
  $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \neg\phi_1$. From definition of $\models_{TR}$ and since ϕ_1 must be an atom, this implies that $\phi_1 \notin M(\langle D_0, \ldots, D_n\rangle)$. We need to prove that $\phi_1 \notin M^{ETR}(\langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle)$, and we do this by contradiction.
  Assume $\phi_1 \in M^{ETR}(\langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle)$. By definition of M^{ETR} this can only be the case if there is a rule in P, s.t. $\phi_1 \leftarrow body$ and $M^{ETR}, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle \models_{ETR} body$. From the proof for the $\Rightarrow$ direction, we know $M^{ETR}, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle \models_{ETR} body$ implies $M, \langle D_0, \ldots, D_n\rangle \models_{TR} body$. Since M is a model of P, then it also models this rule and $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1$ holds. Since ϕ_1 is an atomic formula, this implies that $\phi_1 \in M(\langle D_0, \ldots, D_n\rangle)$ which is impossible. Consequently, $\phi_1 \notin M^{ETR}(\langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle)$ and $M^{ETR}, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle \models_{ETR} \neg\phi_1$

Since P does not have external actions, $M, \pi \rightsquigarrow body$ does not hold for any of the rules in P. As such the first result of this lemma comes directly from Definition 27 and the fact that M is a model of P. □

Definition 28. *Let M be an $\mathcal{ETR}$ interpretation. We define M^{TR} to be the interpretation obtained from M as follows*

1. *If $\phi \in M(\langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle)$ then $\phi \in M^{TR}(\langle D_0, \ldots D_n\rangle)$ for every E (mark that it is always the same E in every state in the path), and every A_i s.t. $A_i \in \mathcal{L}_i$*
2. *nothing else belongs to M^{TR}*

Lemma 2. *Let P be a program without external actions well-formed in both $\mathcal{TR}$ and $\mathcal{ETR}$ and ϕ be a formula without external actions well-formed in both $\mathcal{TR}$ and $\mathcal{ETR}$. Let M be an $\mathcal{ETR}$ model of P and M^{TR} the $\mathcal{TR}$ interpretation obtained by Definition 28. Then:*

M^{TR} is a $\mathcal{TR}$ model of P, and $M, \langle (D_0, E) \xrightarrow{A_1} \ldots \xrightarrow{A_n} (D_n, E)\rangle \models_{ETR} \phi$ iff $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi$

Proof. We start to prove the second claim of this lemma: $M, \langle (D_0, E) \,^{A_1}\!\!\rightarrow \ldots ^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi$ iff
$M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi$ As previously, we divide this proof in two, proving the claim individually for each direction. For the $\Rightarrow$ direction we prove that $M, \langle (D_0, E) \,^{A_1}\!\rightarrow \ldots ^{A_n}\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi$ implies $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi$ and for the $\Leftarrow$ that $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi$ implies $M, \langle (D_0, E) \,^{A_1}\!\!\rightarrow \ldots ^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi$.

$\Rightarrow$:

We prove this by induction on the structure of ϕ.

- *Base Case:* ϕ is an atom
  $M, \langle (D_0, E) \,^{A_1}\!\rightarrow \ldots ^{A_n}\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi$ implies $\phi \in M(\langle (D_0, E) \,^{A_1}\!\rightarrow \ldots ^{A_n}\!\rightarrow (D_n, E)\rangle)$. Then by definition 28 we know that it must be the case that $\phi \in M(\langle D_0, \ldots, D_n\rangle)$ which implies by $\models_{TR}$ $M, \langle (D_0, E) \,^{A_1}\!\rightarrow \ldots ^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{TR} \phi$
- *Conjunction:* $\phi = \phi_1 \wedge \phi_2$
  $M, \langle (D_0, E) \,^{A_1}\!\!\rightarrow \ldots ^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi_1 \wedge \phi_2$, and since this path does not have external actions in its transitions, we know from Theorem 2 that $M, \langle (D_0, E) \,^{A_1}\!\rightarrow \ldots ^{A_n}\!\rightarrow (D_n, E)\rangle \models_c \phi_1 \wedge \phi_2$. Moreover, this is equivalent to say that $M, \langle (D_0, E) \,^{A_1}\!\!\rightarrow \ldots ^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi_1$ and $M, \langle (D_0, E) \,^{A_1}\!\!\rightarrow \ldots ^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi_2$. From this, we can apply our Induction Hypothesis individually to the formulas ϕ_1 and ϕ_2, concluding $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1$ and $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_2$. Finally, by definition of $\models_{TR}$ we know that this is equivalent to $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1 \wedge \phi_2$
- *Disjunction:* $\phi = \phi_1 \vee \phi_2$
  $M, \langle (D_0, E) \,^{A_1}\!\!\rightarrow \ldots ^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi_1 \vee \phi_2$, and since this path does not have external actions in its transitions, we know from Theorem 2 that $M, \langle (D_0, E) \,^{A_1}\!\rightarrow \ldots ^{A_n}\!\rightarrow (D_n, E)\rangle \models_c \phi_1 \vee \phi_2$. Moreover, this is equivalent to say $M, \langle (D_0, E) \,^{A_1}\!\!\rightarrow \ldots ^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi_1$ or $M, \langle (D_0, E) \,^{A_1}\!\rightarrow \ldots ^{A_n}\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi_2$. We can then apply our Induction Hypothesis to ϕ_1 and ϕ_2, concluding $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1$ or $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_2$. By definition of $\models_{TR}$ we know that this is equivalent to $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1 \vee \phi_2$.
- *Serial Conjunction:* $\phi = \phi_1 \otimes \phi_2$
  $M, \langle (D_0, E) \,^{A_1}\!\!\rightarrow \ldots ^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi_1 \otimes \phi_2$, and since this path does not have external actions in its transitions, we know from Theorem 2 that $M, \langle (D_0, E) \,^{A_1}\!\!\rightarrow \ldots ^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_c \phi_1 \otimes \phi_2$. Moreover, this is equivalent to say that there is a i where $0 \leq i \leq n$, such that $M, \langle (D_0, E) \,^{A_1}\!\!\rightarrow \ldots ^{A_n}\!\!\rightarrow (D_i, E)\rangle \models_{ETR} \phi_1$ and $M^{TR}, \langle (D_i, E) \,^{A_1}\!\rightarrow \ldots ^{A_n}\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi_2$. We can then apply our Induction Hypothesis to ϕ_1 and ϕ_2, concluding $M^{TR}, \langle D_0, \ldots, D_i\rangle \models_{TR} \phi_1$ and $M^{TR}, \langle D_i, \ldots, D_n\rangle \models_{TR} \phi_2$. By definition of $\models_{TR}$ we know that this is equivalent to $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1 \otimes \phi_2$.
- *Negation:* $\phi = \neg\phi_1$
  $M, \langle (D_0, E) \,^{A_1}\!\!\rightarrow \ldots ^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{ETR} \neg\phi$, since this path does not have external actions in its transitions, from Theorem 2 we know that $M, \langle (D_0, E) \,^{A_1}\!\rightarrow \ldots ^{A_n}\!\rightarrow (D_n, E)\rangle \models_c \neg\phi_1$. Moreover, since ϕ_1 is an atom, this is equivalent to say $\phi_1 \notin M(\langle (D_0, E) \,^{A_1}\!\rightarrow \ldots ^{A_n}\!\rightarrow (D_n, E)\rangle)$ Then, by definition 28 we know that $\phi_1 \notin M^{TR}(\langle D_0, \ldots, D_n\rangle)$ (as otherwise it would lead to a contradiction). Since ϕ_1 is an atom, $M^{TR}, \pi \models_{TR} \phi_1$ iff $\phi_1 \in M^{TR}(\pi)$. Thus, it is not the case that $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1$ which implies $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \neg\phi_1$.

$\Leftarrow$:

We prove this by induction on the structure of ϕ.

- *Base Case:* ϕ is an atom.
  $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi$ implies that $\phi \in M(\langle D_0, \ldots, D_n\rangle)$. If the latter is true, by definition 28 $\phi \in M(\langle (D_0, E) \,^{A_1}\!\rightarrow \ldots ^{A_n}\!\rightarrow (D_n, E)\rangle)$ which by definition of $\models_{ETR}$ implies that $M, \langle (D_0, E) \,^{A_1}\!\!\rightarrow \ldots ^{A_n}\!\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi$
- *Conjunction:* $\phi = \phi_1 \wedge \phi_2$
  $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1 \wedge \phi_2$. From definition of $\models_{TR}$ this implies $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1$ and $M, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_2$. We are now in conditions to apply our Induction Hypothesis individually to ϕ_1 and ϕ_2 and conclude that $M^{ETR}, \langle (D_0, E) \,^{A_1}\!\rightarrow \ldots ^{A_n}\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi_1$ and $M^{ETR}, \langle (D_0, E) \,^{A_1}\!\rightarrow \ldots ^{A_n}\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi_2$. From this, as intended, we conclude: $M^{ETR}, \langle (D_0, E) \,^{A_1}\!\rightarrow \ldots ^{A_n}\!\rightarrow (D_n, E)\rangle \models_{ETR} \phi_1 \wedge \phi_2$
- *Disjunction:* $\phi = \phi_1 \vee \phi_2$
  Similar to the proof of conjunction, but where $\wedge$ is replaced by $\vee$, and "and" by "or".
- *Serial Conjunction:* $\phi = \phi_1 \otimes \phi_2$
  From $\mathcal{TR}$'s definition of satisfaction, $\models_{TR}$, if $M^{TR}, \langle D_0, \ldots, D_n\rangle \models_{TR} \phi_1 \otimes \phi_2$ then there must exist an i such that $M^{TR}, \langle D_0, \ldots, D_i\rangle \models_{TR}$

ϕ_1 and $M^{TR}, \langle D_i, \ldots, D_n \rangle \models_{TR} \phi_2$ (where $0 \leq i \leq n$). Applying the Induction Hypothesis individually to ϕ_1 and ϕ_2 one concludes that $M, \langle (D_0, E) \overset{A_1}{\rightarrow} \ldots \overset{A_n}{\rightarrow} (D_i, E) \rangle \models_{ETR} \phi_1$ and $M, \langle (D_i, E) \overset{A_1}{\rightarrow} \ldots \overset{A_n}{\rightarrow} (D_n, E) \rangle \models_{ETR} \phi_2$. Thus, given the definition of $\models_{ETR}$: $M, \langle (D_0, E) \overset{A_1}{\rightarrow} \ldots \overset{A_n}{\rightarrow} (D_n, E) \rangle \models_{ETR} \phi_1 \otimes \phi_2$

– *Negation:* $\phi = \neg\phi_1$
$M^{TR}, \langle D_0, \ldots, D_n \rangle \models_{TR} \neg\phi_1$. From definition of $\models_{TR}$ and since ϕ_1 must be an atom, this implies that $\phi_1 \notin M(\langle D_0, \ldots, D_n \rangle)$. By definition 28 this also implies that
$\phi_1 \notin M(\langle (D_0, E) \overset{A_1}{\rightarrow} \ldots \overset{A_n}{\rightarrow} (D_n, E) \rangle)$ (otherwise it would lead to a contradiction). And thus $M, \langle (D_0, E) \overset{A_1}{\rightarrow} \ldots \overset{A_n}{\rightarrow} (D_n, E) \rangle \models_{ETR} \neg\phi_1$

We now prove the first claim: If M is a model of P, then M^{TR} is aslo model of P in $\mathcal{TR}$.
To prove this claim we need to show for every rule $head \leftarrow body$ in P that, whenever $M^{TR}, \pi \models_{TR} body$ then $M^{TR}, \pi \models_{TR} head$, for every path $\pi = \langle D_1, \ldots, D_n \rangle$

By the second claim that was previously proven we know that $M^{TR}, \pi \models_{TR} body$ iff $M, \pi' \models_{ETR} body$, where $\pi' = \langle (D_0, E) \overset{A_1}{\rightarrow} \ldots \overset{A_n}{\rightarrow} (D_n, E) \rangle$. Moreover, since M is a model of P we know that $M, \pi' \models_{ETR} head$ and, since $head$ is an atomic formula, we also know that $head \in M(\langle (D_0, E) \overset{A_1}{\rightarrow} \ldots \overset{A_n}{\rightarrow} (D_n, E) \rangle)$. By definition 28 this implies $head \in M^{TR}(\pi)$ and $M, \pi \models \phi$. □

B.2. Auxiliary Results for Theorem 4

Theorem 4 establishes the Soundness and Completeness of the derivation procedure $\vdash$.

B.2.1. Soundness

The auxiliary results up to Lemma 9 contribute to the soundness proof. We start by proving the soundness of $\vdash_c$ w.r.t. to classical satisfaction; then show that replacing in resolvents, an atom which is the head of some rule by the rule's body is a sound operation; then that action failed derivations corresponds to finding formulas that are partial but not classically satisfied, and that rule-5 of the procedure builds compensations. Finally we prove that each of the rules of the procedure is sound w.r.t. the general executional entailment.

Lemma 3 (Soundness $\vdash_c$)**.** *Let P be a serial-Horn program, M be a model of P, π be a path, ϕ a serial goal.*

$$\textit{If } P, \pi \vdash_c \phi \textit{ then } M, \pi \models_c \phi$$

Proof.
Soundness of Axiom:
$P, \pi \vdash_c ()$ only holds for paths π with size 1. Since $()$ represents the empty transaction, which is tautologically true in any path of length 1, the result follows trivially, and $M, \pi \models_c ()$.
Soundness of Rules:
We prove each of the rules, r_1–r_4, separately:

r.1 Assume there is a rule $L_1 \leftarrow B_1 \otimes \ldots \otimes B_j$ in P. We want to prove that if $M, \pi \models_c B_1 \otimes \ldots \otimes B_j \otimes L_2 \otimes \ldots \otimes L_k$ the $M, \pi \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$
Since we have $M, \pi \models_c B_1 \otimes \ldots \otimes B_j \otimes L_2 \otimes \ldots \otimes L_k$ then we know that there is a split $\pi_1 \circ \pi_2$ of π s.t. $M, \pi_1 \models_c B_1 \otimes \ldots \otimes B_j$ and $M, \pi_2 \models_c L_2 \otimes \ldots \otimes L_k$. Then, since M is a model of P and $L_1 \leftarrow B_1 \otimes \ldots \otimes B_j$ is a serial-Horn rule, it follows that $M, \pi_1 \models_c L_1$ and thus $M, \pi \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$

r.2 Assume that $\mathcal{O}^d(D_1) \models L_1$. We want to prove that if $M, \langle (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_f}{\rightarrow} (D_f, E_f) \rangle \models_c L_2 \otimes \ldots \otimes L_k$ then $M, \langle (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_f}{\rightarrow} (D_f, E_f) \rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$ holds.
Since $\mathcal{O}^d(D_1) \models L_1$, we know that for every interpretation, $M, \langle (D_1, E_1) \rangle \models_c L_1$. Thus, by definition of the serial conjunction case of $\models_c$:
$M, \langle (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_f}{\rightarrow} (D_f, E_f) \rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$

r.3 Assume that $\mathcal{O}^t(D_0, D_1) \models L_1$. We want to prove that if $M, \langle (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_f}{\rightarrow} (D_f, E_f) \rangle \models_c L_2 \otimes \ldots \otimes L_k$ then $M, \langle (D_0, E_1) \overset{L_1}{\rightarrow} (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_f}{\rightarrow} (D_f, E_f) \rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$
Since $\mathcal{O}^t(D_0, D_1) \models L_1$, we know that:
$M, \langle (D_0, E_1) \overset{L_1}{\rightarrow} (D_1, E_1) \rangle \models_c L_1$. Thus, by definition of the serial conjunction case of $\models_c$:
$M, \langle (D_0, E_1) \overset{L_1}{\rightarrow} (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_f}{\rightarrow} (D_f, E_f) \rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$

r.4 Assume $\mathcal{O}^e(E_0, E_1) \models L_1$. We want to prove that if $M, \langle (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_f}{\rightarrow} (D_f, E_f) \rangle \models_c L_2 \otimes \ldots \otimes L_k$ then $M, \langle (D_1, E_0) \overset{L_1}{\rightarrow} (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_f}{\rightarrow} (D_f, E_f) \rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$
Since $\mathcal{O}^e(E_0, E_1) \models L_1$, we know:
$M, \langle (D_1, E_0) \overset{L_1}{\rightarrow} (D_1, E_1) \rangle \models_c L_1$. Thus, by definition of the serial conjunction case of $\models_c$:
$M, \langle (D_1, E_0) \overset{L_1}{\rightarrow} (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_f}{\rightarrow} (D_f, E_f) \rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$

□

Definition 29 (Unfolding of formulas)**.** *Let P be a serial-Horn program, and ϕ a serial-goal of the form: $\phi = \phi_1 \otimes \ldots \otimes \phi_i \otimes \ldots \otimes \phi_k$. A one-step unfolding of ϕ is a serial goal obtained from ϕ by replacing one atom ϕ_i ($1 \leq i \leq k$) in ϕ by a serial goal body, in case $\phi_i \leftarrow$ body is a rule in P. An unfolding of ϕ is a serial goal obtained from ϕ by iteratively applying one-step unfolding a finite number of times.*

A serial-Horn goal is completely unfolded if it is empty, or if every atom in it is an action formula defined in the oracles.

Lemma 4. *Let ϕ be a completely unfolded serial-Horn goal formula, and M an interpretation s.t. $M \models P$.*

If $M, \pi \models_p \phi$ and $M, \pi \not\models_c \phi$, then $\forall \psi$ s.t. ψ is a serial-Horn goal, it is the case that $M, \pi \models_p \phi \otimes \psi$ and $M, \pi \not\models_c \phi \otimes \psi$

Proof. We prove this by induction on the size of the serial-Horn goal $\phi = \phi_1 \otimes \ldots \otimes \phi_k$:

Inductive Hypothesis: Let π be a path, and M a model of a program P. For completely unfolded serial goals ψ and $\phi_1 \otimes \ldots \otimes \phi_k$:

If $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_k$ and $M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_k$ then $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_k \otimes \psi$ and $M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_k \otimes \psi$

Base, k = 1:

Let ϕ be a serial-Horn formula of size 1. Since ϕ is an atom, $M, \pi \models_p \phi \wedge M, \pi \not\models_c \phi$ only if π is a 1-path. By definition of $\models_p$, it follows immediately that $M, \pi \models_p \phi \otimes \psi$. Moreover, since π is a 1-path, and $M, \pi \not\models_c \phi$ then by definition of the serial conjunction $\otimes$ in $\models_c$ the only split that can be done from π is $\pi = \pi \circ \pi$ and thus it follows that $M, \pi \not\models_c \phi \otimes \psi$.

Induction Step:

Assume our hypothesis is true for values up to k. We prove that it is still true for $\phi_1 \otimes \ldots \otimes \phi_{k+1}$.

Assume that $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_{k+1} \wedge M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_{k+1}$. By definition of serial conjunction in $\models_p$, this is equivalent to: $[(M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_k \wedge M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_k) \vee (\exists \pi_1 \circ \pi_2 = \pi \; s.t. \; M, \pi_1 \models_c \phi_1 \otimes \ldots \otimes \phi_k \wedge M, \pi_2 \models_p \phi_{k+1})] \wedge \forall \pi_3 \circ \pi_4 = \pi. \; M, \pi_3 \not\models_c \phi_1 \otimes \ldots \otimes \phi_k \vee M, \pi_4 \not\models_c \phi_{k+1}$

We consider each of these two cases individually:

(1) Assume $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_k \wedge M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_k$ is true. By induction hypothesis we know that $\forall \psi_1$ s.t. ψ_1 is a serial-Horn goal, then $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_k \otimes \psi_1$ and $M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_k \otimes \psi_1$. Moreover, since this holds for any formula ψ_1 s.t. ψ_1 is a serial-Horn goal, then it is also true for $\psi_1 = \phi_{k+1} \otimes \psi$ and thus $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_{k+1} \otimes \psi$ and $M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_{k+1} \otimes \psi$.

(2) Assume that $\exists \pi_1 \circ \pi_2 = \pi \; s.t. \; M, \pi_1 \models_c \phi_1 \otimes \ldots \otimes \phi_k \wedge M, \pi_2 \models_p \phi_{k+1}$ and that $\forall \pi_3 \circ \pi_4 = \pi. \; M, \pi_3 \not\models_c \phi_1 \otimes \ldots \otimes \phi_k \vee M, \pi_4 \not\models_c \phi_{k+1}$.

Since $M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_{k+1}$ and $M, \pi_1 \models_c \phi_1 \otimes \ldots \otimes \phi_k$, it follows that $M, \pi_2 \models_p \phi_{k+1}$ and $M, \pi_2 \not\models_c \phi_{k+1}$. Since ϕ_{k+1} is a completely unfolded serial-Horn formula of size 1, we are in the base case that was already proven, and we can conclude that $M, \pi_2 \not\models_c \phi_{k+1} \otimes \psi$ and $M, \pi_2 \models_p \phi_{k+1} \otimes \psi$. Then by definition of $\models_p$ we know that $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_k \otimes \psi$ Moreover, recall from the base case proof, π_2 must be a 1-path, and by definition of split, $\pi_1 = \pi$. Since $M, \pi \models_c \phi_1 \otimes \ldots \otimes \phi_k$ and $M, \pi_2 \not\models_c \phi_{k+1} \otimes \psi$ then $M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_k \otimes \phi_{k+1} \otimes \psi$ □

Lemma 5. *Let M be an interpretation, ϕ any transaction formula, ψ an atomic action defined in the oracles and π a path s.t. $\pi = \pi_1 \circ \pi_2$. If $M, \pi_1 \models_c \psi$ and $M, \pi_2 \not\models_c \psi$ then $M, \pi \not\models_c \psi \otimes \phi$*

Proof. We make this proof by contradiction. Assume $M, \pi \models_c \psi \otimes \phi$ holds. Then there is a split π_3, π_4 s.t. $M, \pi_3 \models_c \psi$ and $M, \pi_4 \models_c \phi$. Clearly, this property only holds if $\pi_4 \neq \pi_2$ which implies that $\pi_1 \neq \pi_3$

Since ψ is an atom, for any path π' $M, \pi' \models_c \psi$ iff $\psi \in M(\pi')$. Since ψ is an *action defined in the oracles*, by definition of interpretation (Definition 11) we know that this π' must be a 2-path. Since π' must be a 2-path to satisfy ψ, and π_1 and π_3 must always be a prefix of π (by the definition of path split), we can conclude that $\pi_1 = \pi_3$ and thus that $\pi_2 = \pi_4$. Consequently, $M, \pi_4 \not\models_c \phi$,and the assumption is contradicted. □

Lemma 6 (Soundness of action-failed derivation w.r.t. $\models_c$ and $\models_p$)**.** *If there is an action-failed derivation starting in $\langle S_1 \rangle, S_1 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_i$ and ending in $\langle S_1 \, {}^{A_1}\!\!\rightarrow \ldots {}^{A_{f-1}}\!\!\rightarrow S_f \rangle, S_f \Vdash_P \psi$, for some serial-goal ψ, then for all models M of P, $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_k$ and $M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_k$, for $\pi = \langle S_1 \, {}^{A_1}\!\!\rightarrow \ldots {}^{A_{f-1}}\!\!\rightarrow S_f \rangle$*

Proof. We start by proving by induction on the size k of the serial-Horn formula $\phi_1 \otimes \ldots \otimes \phi_k$ that:

Inductive Hypothesis: If there is an action-failed derivation starting in $\langle S_1 \rangle, S_1 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_i$ and ending in $\langle S_1 \, {}^{A_1}\!\!\rightarrow \ldots {}^{A_{f-1}}\!\!\rightarrow S_f \rangle, S_f \Vdash_P \psi$, for some serial-goal ψ, then $\forall M$ s.t. $M \models P$, $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_k$ and $M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_k$ for $\pi = \langle S_1 \, {}^{A_1}\!\!\rightarrow \ldots {}^{A_{f-1}}\!\!\rightarrow S_f \rangle$

Base Case $k = 1$
If $k = 1$, then, by definition of action-failed derivation, π must be a 1-path, $\langle S_f \rangle = \langle (D_f, E_f) \rangle$, the derivation starts and ends in the following resolvent: $\langle (D_f, E_f) \rangle, (D_f, E_f) \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$, and one of the two cases must occur:

(i). $\phi_1 \in \mathcal{L}_i$, $\mathcal{O}^d(D_f) \not\models \phi_1$ and $\neg\exists D_i$ s.t. $\mathcal{O}^t(D_f, D_i) \models \phi_1$, or
(ii). $\phi_1 \in \mathcal{L}_a^*$ and $\neg\exists E_i$ s.t. $\mathcal{O}^e(E_f, E_i) \models \phi_1$

Since ϕ_1 belongs to the language of the oracles, then it means that, for any M, we know that $\phi_1 \notin M(\langle (D_f, E_f) \rangle)$; but also, that $\neg\exists D_i, E_f$ such that $\phi_1 \in M\langle (D_f, E_f) \overset{\phi_1}{\rightarrow} (D_i, E_f) \rangle$ or that $\phi_1 \in M\langle (D_f, E_f) \overset{\phi_1}{\rightarrow} (D_f, E_i) \rangle$. As a result, by the definitions of $\models_p$ and $\models_c$, we have $M, \langle (D_f, E_f) \rangle \not\models_c \phi_1$, and $M, \langle (D_f, E_f) \rangle \models_p \phi_1$.

Induction Step:
Assume there is an action-failed derivation starting in $\langle S_1 \rangle, S_1 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_{k+1}$ and ending in $\langle S_1 \overset{A_1}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} S_f \rangle, S_f \Vdash_P \psi$, for some serial-goal ψ. Since $\phi_1 \otimes \ldots \otimes \phi_{k+1}$ is a completely unfolded formula, ψ must be a sub formula of $\phi_1 \otimes \ldots \otimes \phi_{k+1}$.

More precisely, there must exist a resolvent: $\langle S_1, \ldots, S_f \rangle, S_f \Vdash_P \phi_i \otimes \ldots \otimes \phi_{k+1}, (1 \leq i \leq k+1)$ where one of the following conditions are true:

(i). $\phi_i \in \mathcal{L}_i$, $\mathcal{O}^d(D_f) \not\models L_1$ and $\neg\exists D_i$ s.t. $\mathcal{O}^t(D_f, D_i) \models \phi_i$, or
(ii). $\phi_i \in \mathcal{L}_a^*$ and $\neg\exists E_i$ s.t. $\mathcal{O}^e(E_f, E_i) \models \phi_i$

In this case there are two possibilities:

1. $(1 \leq i \leq k)$ and by definition there is a classical derivation starting in $\langle S_1 \rangle, S_1 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$ and ending in $\pi, S_f \Vdash_P ()$. Since $i \leq k$, by induction hypothesis we know that $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_i$ and $M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_i$ and by Lemma 4 we know that $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_i \otimes \ldots \otimes \phi_{k+1}$ and $M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_i \otimes \ldots \otimes \phi_{k+1}$ for all $\phi_{i+1} \otimes \ldots \phi_{k+1}$
2. $(i = k+1)$ and by definition there is a classical derivation starting in $\langle S_1 \rangle, S_1 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$ and ending in $\pi, S_f \Vdash_P ()$. If so, we know, by Lemma 3, that $M, \pi \models_c \phi_1 \otimes \ldots \otimes \phi_k$. Moreover, we also have an action-failed derivation starting in $\langle S_f \rangle, S_f \Vdash_P \phi_{k+1}$ and ending in $\langle S_f \rangle, S_f \Vdash_P \phi_{k+1}$. Since ϕ_{k+1} is of size 1, as proven in the base case, we know $M, \langle S_f \rangle \models_p \phi_{k+1}$ and $M, \langle S_f \rangle \not\models_c \phi_{k+1}$. Since $\pi = \langle S_1 \overset{A_1}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} S_f \rangle$ we can conclude that $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_k \otimes \phi_{k+1}$ and $M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_k \otimes \phi_{k+1}$.

Soundness of Rules
To prove the soundness of rules we need to prove the following:

r.1 Assume there is a rule $\phi_1 \leftarrow \psi \in P$. If $M, \pi \models_p \psi \otimes \phi_2 \otimes \ldots \otimes \phi_k$ and $M, \pi \not\models_c \psi \otimes \phi_2 \otimes \ldots \otimes \phi_k$ then $M, \pi \models_p \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$ and $M, \pi \not\models_c \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$
r.2 Assume $\mathcal{O}^d(D_1) \models \phi_1$
If $M, \pi \not\models_c \phi_2 \otimes \ldots \otimes \phi_k$ and $M, \pi \models_p \phi_2 \otimes \ldots \otimes \phi_k$ then it holds:
$M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_k$ and $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_k$
r.3 Assume $\mathcal{O}^d(D_1, D_2) \models L_1$
If $M, \langle (D_2, E_2) \overset{A_2}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} (D_f, E_f) \rangle \not\models_c \phi_2 \otimes \ldots \otimes \phi_k$ and $M, \langle (D_2, E_2) \overset{A_2}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} (D_f, E_f) \rangle \models_p \phi_2 \otimes \ldots \otimes \phi_k$ then it holds:
$M, \langle (D_1, E_2) \overset{\phi_1}{\rightarrow} (D_2, E_2) \overset{A_2}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} (D_f, E_f) \rangle \not\models_c \phi_1 \otimes \ldots \otimes \phi_k$ and $M, \langle (D_1, E_2) \overset{\phi_1}{\rightarrow} (D_2, E_2) \overset{A_2}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} (D_f, E_f) \rangle \models_p \phi_1 \otimes \ldots \otimes \phi_k$
r.4 Assume $\mathcal{O}^e(E_0, E_1) \models \phi_1$
If $M, \langle (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} (D_f, E_f) \rangle \not\models_c \phi_2 \otimes \ldots \otimes \phi_k$ and $M, \langle (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} (D_f, E_f) \rangle \models_p \phi_2 \otimes \ldots \otimes \phi_k$ then it holds:
$M, \langle (D_1, E_0) \overset{\phi_1}{\rightarrow} (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} (D_f, E_f) \rangle \not\models_c \phi_1 \otimes \ldots \otimes \phi_k$ and $M, \langle (D_1, E_0) \overset{\phi_1}{\rightarrow} (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} (D_f, E_f) \rangle \models_p \phi_1 \otimes \ldots \otimes \phi_k$

We prove each rule individually:

r.1 Not applicable since $\phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$ is a completely unfolded goal
r.2 Assume $\mathcal{O}^d(D_1) \models \phi_1$, $M, \langle (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} (D_f, E_f) \rangle \not\models_c \phi_2 \otimes \ldots \otimes \phi_k$ and $M, \langle (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} (D_f, E_f) \rangle \models_p \phi_2 \otimes \ldots \otimes \phi_k$.
Since $\mathcal{O}^d(D_1) \models \phi_1$, $\phi_1 \in M(\langle (D_1, E_1) \rangle)$ holds, by the serial conjunction case of the partial satisfaction definition, we can conclude that $M, \langle (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} (D_f, E_f) \rangle \models_p \phi_1 \otimes \ldots \otimes \phi_k$. Moreover, since ϕ_1 is an oracle primitive, we can apply Lemma 5 and conclude that: $M, \langle (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} (D_f, E_f) \rangle \not\models_c \phi_1 \otimes \ldots \otimes \phi_k$
r.3 Assume $\mathcal{O}^t(D_0, D_1) \models \phi_1$, $M, \langle (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} (D_f, E_f) \rangle \not\models_c \phi_2 \otimes \ldots \otimes \phi_k$ and $M, \langle (D_1, E_1) \overset{A_1}{\rightarrow} \ldots \overset{A_{f-1}}{\rightarrow} (D_f, E_f) \rangle \models_p \phi_2 \otimes \ldots \otimes \phi_k$
Since $\mathcal{O}^t(D_0, D_1) \models \phi_1$, we know that $\phi_1 \in M(\langle (D_0, E_1) \overset{L_1}{\rightarrow} (D_1, E_1) \rangle)$. Thus, by the serial conjunction case of the partial satisfaction defini-

tion we can conclude: $M, \langle (D_0, E_1)\ ^{L_1}\!\!\rightarrow (D_1, E_1)\ ^{A_1}\!\!\rightarrow \ldots\ ^{A_{f-1}}\!\!\rightarrow (D_f, E_f)\rangle \models_p \phi_1 \otimes \ldots \otimes \phi_k$. Moreover, since ϕ_1 is an oracle primitive, we can apply Lemma 5 and conclude: $M, \langle (D_0, E_1)\ ^{L_1}\!\!\rightarrow (D_1, E_1)\ ^{A_1}\!\!\rightarrow \ldots\ ^{A_{f-1}}\!\!\rightarrow (D_f, E_f)\rangle \not\models_c \phi_1 \otimes \ldots \otimes \phi_k$

r.4 Assume $\mathcal{O}^e(E_0, E_1) \models \phi_1$, $M, \langle (D_1, E_1)\ ^{A_1} \rightarrow \ldots\ ^{A_{f-1}} \rightarrow (D_f, E_f)\rangle \not\models_c \phi_2 \otimes \ldots \otimes \phi_k$ and $M, \langle (D_1, E_1)\ ^{A_1}\!\!\rightarrow \ldots\ ^{A_{f-1}}\!\!\rightarrow (D_f, E_f)\rangle \models_p \phi_2 \otimes \ldots \otimes \phi_k$
Since $\mathcal{O}^e(E_0, E_1) \models \phi_1$ we know: $\phi_1 \in M(\langle (D_1, E_0)\ ^{\phi_1} \rightarrow (D_1, E_1)\rangle)$. Thus, by the serial conjunction case of the partial satisfaction definition we can conclude that $M, \langle (D_1, E_0)\ ^{L_1} \rightarrow (D_1, E_1)\ ^{A_1} \rightarrow \ldots\ ^{A_{f-1}} \rightarrow (D_f, E_f)\rangle \models_p \phi_1 \otimes \ldots \otimes \phi_k$. Moreover, since ϕ_1 is an oracle primitive, we can apply Lemma 5 and conclude: $M, \langle (D_1, E_0)\ ^{L_1} \rightarrow (D_1, E_1)\ ^{A_1} \rightarrow \ldots\ ^{A_{f-1}} \rightarrow (D_f, E_f)\rangle \not\models_c \phi_1 \otimes \ldots \otimes \phi_k$

□

Lemma 7 (Soundness of rule-5. w.r.t. $\rightsquigarrow$)**.** *Let P be a serial-Horn program, $\phi_1 \otimes \ldots \otimes \phi_k$ be a completely unfolded serial-Horn goal.*

If all the following conditions are true:

1. *There is an action-failed classical derivation starting in $\langle S_1\rangle, S_1 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$ (where $S_1 = (D_1, E_1)$) ending in $\langle S_1\ ^{A_1}\!\!\rightarrow \ldots\ ^{A_{j-1}}\!\!\rightarrow S_j\rangle, S_j \Vdash_P \psi$, for some serial-goal ψ*
2. *$S_1\ ^{A_1}\!\!\rightarrow \ldots\ ^{A_{p-1}}\!\!\rightarrow S_p$ is the rollback path of the path $S_1\ ^{A_1}\!\!\rightarrow \ldots\ ^{A_{j-1}}\!\!\rightarrow S_j$ (cf. Definition 15)*
3. *$\mathtt{Inv}(\mathtt{Seq}(\langle S_1\ ^{A_1}\!\!\rightarrow \ldots\ ^{A_{p-1}}\!\!\rightarrow S_p\rangle)) = A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ (cf. Definition 16)*
4. *$P, \langle S_p\ ^{A_k^{-1}}\!\!\rightarrow \ldots\ ^{A_1^{-1}}\!\!\rightarrow S_q\rangle, S_q \vdash_c A_k^{-1} \otimes \ldots \otimes A_1^{-1}$*

then $M, \langle S_1\ ^{A_1}\!\!\rightarrow \ldots\ ^{A_{p-1}}\!\!\rightarrow S_p\ ^{A_{p-1}^{-1}}\!\!\rightarrow \ldots\ ^{A_1^{-1}}\!\!\rightarrow S_q\rangle \rightsquigarrow \phi_1 \otimes \ldots \otimes \phi_k$ for all models M of P

Proof. From Lemma 6 we know that if $\phi_1 \otimes \ldots \otimes \phi_k$ is a completely unfolded serial-Horn goal and there is an action-failed classical derivation starting in $\langle S_1\rangle, S_1 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$ (where $S_1 = (D_1, E_1)$) ending in $\langle S_1\ ^{A_1}\!\!\rightarrow \ldots\ ^{A_{j-1}}\!\!\rightarrow S_j\rangle, S_j \Vdash_P \psi$, for some serial-goal ψ then $M, \langle S_1\ ^{A_1} \rightarrow \ldots\ ^{A_{j-1}} \rightarrow S_j\rangle \models_p \phi_1 \otimes \ldots \otimes \phi_k$ and $M, \langle S_1\ ^{A_1} \rightarrow \ldots\ ^{A_{j-1}} \rightarrow S_j\rangle \not\models_c \phi_1 \otimes \ldots \otimes \phi_k$.

Note that $S_1\ ^{A_1}\!\!\rightarrow \ldots\ ^{A_{p-1}}\!\!\rightarrow S_p$ is the rollback path of the path $S_1\ ^{A_1}\!\!\rightarrow \ldots\ ^{A_{j-1}}\!\!\rightarrow S_j$ (applying Definition 15) and $\mathtt{Inv}(\mathtt{Seq}(\langle S_1\ ^{A_1}\!\!\rightarrow \ldots\ ^{A_{p-1}}\!\!\rightarrow S_p\rangle)) = A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ (applying Definition 16).

By Lemma 3 we know that: $P, \langle S_p\ ^{A_k^{-1}}\!\!\rightarrow \ldots\ ^{A_1^{-1}}\!\!\rightarrow S_q\rangle, S_q \vdash_c A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ then $M, \langle S_p\ ^{A_k^{-1}}\!\!\rightarrow \ldots\ ^{A_1^{-1}}\!\!\rightarrow S_q\rangle, S_q \models_c A_k^{-1} \otimes \ldots \otimes A_1^{-1}$.

Then, by Definition 17 we can conclude that if conditions 1-4 hold then: $M, \langle S_1\ ^{A_1} \rightarrow \ldots\ ^{A_{p-1}} \rightarrow S_p\ ^{A_{p-1}^{-1}}\!\!\rightarrow \ldots\ ^{A_1^{-1}}\!\!\rightarrow S_q\rangle \rightsquigarrow \phi_1 \otimes \ldots \otimes \phi_k$ □

Definition 30. *M_{def} is an interpretation defined w.r.t. a program P as follows:*

- *$\phi \in M_{def}(\pi)$ and $\phi \in \mathcal{L}_{\mathcal{O}}$ iff $\mathcal{O}^d(\pi) \models \phi$ or $\mathcal{O}^t(\pi) \models \phi$ or $\mathcal{O}^e(\pi) \models \phi$.*
- *$\phi \in M_{def}(\pi)$ and $\phi \in \mathcal{L}_P$ iff there is a rule $\phi \leftarrow body \in P$ and $M, \pi \models body$*
- *$M_{def}(\pi) \rightsquigarrow \phi$ and ϕ is a complex formula iff conditions 1-4 of Definition 17 are true.*
- *$M_{def}(\pi) \rightsquigarrow \phi$ and ϕ is an atom iff there is a rule $\phi \leftarrow body \in P$ and $M, \pi \rightsquigarrow body$*

Remark 1. *It follows easily from the definition, that M_{def} is a valid $\mathcal{ETR}$ interpretation and a model of program P.*

Lemma 8. *Let P be a program containing the rule $\psi \leftarrow \phi_1$. For every model M of P:*

If $M, \pi \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$ then $M, \pi \models \psi \otimes \phi_2 \otimes \ldots \otimes \phi_j$.

Proof. We will apply induction on the size k of path $\pi = \langle S_1\ ^{A_1}\!\!\rightarrow \ldots\ ^{A_{n-1}}\!\!\rightarrow, S_k\rangle$

Induction Hypothesis: For every model M of P where P contains the rule $\psi \leftarrow \phi_1$, if $M, \pi \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$ then $M, \pi \models \psi \otimes \phi_2 \otimes \ldots \otimes \phi_j$.

Base

If $k = 1$, since it is impossible to construct a compensating path that has size 1, $M, \pi \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$ is only true by the serial conjunction case. Moreover, since π has size 1, $M, \pi \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_1$ iff $M, \pi \models \phi_1$, $M, \pi \models \phi_2$, ..., $M, \pi \models \phi_j$. Since for every path π_1: if $M, \pi_1 \models \phi_1$ then, since M is a model, we have $M, \pi_1 \models \psi$. Since $M, \pi \models \psi$, and $M, \pi \models \phi_2 \otimes \ldots \otimes \phi_j$, and so we also have $M, \pi \models \psi \otimes \phi_2 \otimes \ldots \otimes \phi_j$.

Step

Let's assume that the hypothesis is true for paths up to size j. Here we prove that it is also true for paths of size $j + 1$.

By definition of $\models$, we know that $M, \pi \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$ holds if either of the cases occur:

(a) Serial Conjunction Case: $M, \pi \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$ is true because there is a split $\pi_1 \circ \pi_2$ of π s.t. $M, \pi_1 \models \phi_1$ and $M, \pi_2 \models \phi_2 \otimes \ldots \otimes \phi_j$. Since

M is a model of P, we know $M, \pi \models \psi$ and then by the definition of Serial Conjunction, $M, \pi \models \psi \otimes \phi_2 \otimes \ldots \otimes \phi_j$.

(b) Compensating Case: $M, \pi \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$ is true because there is a split $\pi_1 \circ \pi_2$ of π s.t. $M, \pi_1 \rightsquigarrow \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$ and $M, \pi_2 \models \phi \otimes \phi_2 \otimes \ldots \otimes \phi_j$.
In this case π_2 is strictly smaller than π as otherwise π_1 would be a 1-path and $M, \pi_1 \rightsquigarrow \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$ would not hold. Consequently, we can apply the Induction Hypothesis to π_2 and conclude $M, \pi_2 \models \psi \otimes \phi_2 \otimes \ldots \otimes \phi_j$.
If $M, \pi_1 \rightsquigarrow \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$, there is a path π' s.t. $M, \pi' \models_p \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$ and $M, \pi' \not\models_c \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$. By definition of the serial conjunction case in the partial satisfaction relation we know that:

(b1) $M, \pi_1 \rightsquigarrow \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$ because rules 1-4 of Definition 17 hold, and the failure occurs in ϕ_1, i.e., $M, \pi' \models_p \phi_1$ and $M, \pi' \not\models_c \phi_1$. Since M is a model of program, we know $M, \pi_1 \rightsquigarrow \phi_1$ implies $M, \pi_1 \rightsquigarrow \psi$ and thus $M, \pi_1 \rightsquigarrow \psi \otimes \phi_2 \otimes \ldots \otimes \phi_j$. Since $M, \pi_2 \models \psi \otimes \phi_2 \otimes \ldots \otimes \phi_j$ we can conclude $M, \pi \models \psi \otimes \phi_2 \otimes \ldots \otimes \phi_j$.

(b2) $M, \pi_1 \rightsquigarrow \phi \otimes \phi_2 \otimes \ldots \otimes \phi_j$ because rules 1-4 of Definition 17 hold, and the failure occurs after ϕ_1. I.e., there is a path π' with a split $\pi'_1 \circ \pi'_2$ of π' s.t. $M, \pi'_1 \models_c \phi_1$, $M, \pi'_2 \models_p \phi_2 \otimes \ldots \otimes \phi_j$ and $M, \pi'_2 \not\models_c \phi_2 \otimes \ldots \otimes \phi_j$. Since M is a model of P and $M, \pi'_1 \models_c \phi_1$ it holds $M, \pi'_1 \models_c \psi$, and thus $M, \pi_1 \rightsquigarrow \psi \otimes \phi_2 \otimes \ldots \otimes \phi_j$. From this and $M, \pi_2 \models \psi \otimes \phi_2 \otimes \ldots \otimes \phi_j$ we know that $M, \pi \models \phi \otimes \phi_2 \otimes \ldots \otimes \phi_j$

(b3) $M, \pi_1 \rightsquigarrow \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$ and rules 1-4 of Definition 17 do not hold. However, this must be true for every model M of P, and thus it must hold also for model M_{def} specified in Definition 30. By M_{def} definition, $M_{def}, \pi_1 \rightsquigarrow \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$ and rules 1-4 of Definition 17 do not hold, only if $\phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j$ is an atom, i.e. if $\phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_j = \phi_1$. If this is the case, then by definition of what is a model, $M, \pi_1 \rightsquigarrow \phi_1$ implies $M, \pi \rightsquigarrow \psi$ and thus $M, \pi \models \psi$ as intended.

□

Lemma 9 (Soundness $\Vdash_P$)**.** *Let P be a serial-Horn program, $\phi_1 \otimes \ldots \otimes \phi_k$ ($k \geq 1$) be a serial-Horn goal and π_1,π_2 be paths. Let $\pi_1 \circ \pi_2, S_j \Vdash_P \psi$ be the next derivation step of $\pi_1, S_i \Vdash_P \phi$.*
If $P, \pi_1 \models \phi$ then $P, \pi_1 \circ \pi_2 \models \psi$

Proof. We have to prove the following:

r.1 Assume $\pi, S_i \Vdash_P \psi \otimes \phi_2 \otimes \ldots \otimes \phi_k$, $\pi, S_i \Vdash_P \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$ and $\phi_1 \leftarrow \psi$ is a rule in P.
If $P, \pi \models \psi \otimes \phi_2 \otimes \ldots \otimes \phi_k$ then $P, \pi \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$

r.2 Assume $\pi, (D_i, E_i) \Vdash_P \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$, $\pi, (D_i, E_i) \Vdash_P \phi_2 \otimes \ldots \otimes \phi_k$, $\mathcal{O}^d(D_1) \models \phi_1$, and $\pi = \langle (D_1, E_1) \xrightarrow{A_1} \ldots \xrightarrow{A_{i-1}} (D_i, E_i) \rangle$
If $P, \pi \models \phi_2 \otimes \ldots \otimes \phi_k$ then $P, \pi \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$

r.3 Assume $\pi, (D_i, E_i) \Vdash_P \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$, $\pi, (D_i, E_i) \Vdash_P \phi_2 \otimes \ldots \otimes \phi_k$, $\mathcal{O}^t(D_0, D_1) \models \phi_1$, and $\pi = \langle (D_1, E_1) \xrightarrow{A_1} \ldots \xrightarrow{A_{i-1}} (D_i, E_i) \rangle$
If $P, \pi \models \phi_2 \otimes \ldots \otimes \phi_k$ then $P, \langle (D_0, E_1) \xrightarrow{\phi_1} (D_1, E_1) \rangle \circ \pi \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$

r.4 Assume $\pi, (D_i, E_i) \Vdash_P \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$, $\pi, (D_i, E_i) \Vdash_P \phi_2 \otimes \ldots \otimes \phi_k$, $\mathcal{O}^e(E_0, E_1) \models \phi_1$, and $\pi = \langle (D_1, E_1) \xrightarrow{A_1} \ldots \xrightarrow{A_{i-1}} (D_i, E_i) \rangle$
If $P, \pi \models \phi_2 \otimes \ldots \otimes \phi_k$ then $P, \langle (D_1, E_0) \xrightarrow{\phi_1} (D_1, E_1) \rangle \circ \pi \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$

r.5 Assume $\langle S_q \xrightarrow{A_q} \ldots \xrightarrow{A_{i-1}} S_i \rangle, (D_i, E_i) \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$ and $\langle S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{p-1}} S_p \xrightarrow{A_{p-1}^{-1}} \ldots \xrightarrow{A_1^{-1}} S_q \xrightarrow{A_q} \ldots \xrightarrow{A_{i-1}} S_i \rangle, (D_i, E_i) \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$ and all the following conditions hold:

(a) There is an action-failed derivation starting in $\langle S_1 \rangle, S_1 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$ and ending in $\langle S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{j-1}} S_j \rangle, S_j \Vdash_P \psi$, for some serial-goal ψ

(b) $S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{p-1}} S_p$ is the rollback path of the path $S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{j-1}} S_j$ (cf. Definition 15)

(c) $\texttt{Inv}(\texttt{Seq}(\langle S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{p-1}} S_p \rangle)) = A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ (cf. Definition 16)

(d) $P, \langle S_p \xrightarrow{A_k^{-1}} \ldots \xrightarrow{A_1^{-1}} S_q \rangle, S_q \vdash_c A_k^{-1} \otimes \ldots \otimes A_1^{-1}$

If $P, \langle S_q \xrightarrow{A_q} \ldots \xrightarrow{A_{i-1}} S_i \rangle \models \phi_1 \otimes \ldots \otimes \phi_k$ then $P, \langle S_1 \xrightarrow{A_1} \ldots \xrightarrow{A_{p-1}} S_p \xrightarrow{A_{p-1}^{-1}} \ldots \xrightarrow{A_1^{-1}} S_q \xrightarrow{A_q} \ldots \xrightarrow{A_{i-1}} S_i \rangle \models \phi_1 \otimes \ldots \otimes \phi_k$

We prove each item in turn.

r.1 We know $P, \pi \models \psi \otimes \phi_2 \otimes \ldots \otimes \phi_k$ and $\phi_1 \leftarrow \psi$ is a rule in P. By definition, this is equivalent to saying: for every model M of P, $M, \pi \models \psi \otimes \phi_2 \otimes \ldots \otimes \phi_k$. Then, by Lemma 8, we know that $M, \pi \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$, and thus $P, \pi \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$

r.2 Assume that $\mathcal{O}^d(D_1) \models \phi_1$ and $P, \langle (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ holds. By definition of $\mathcal{ETR}$ interpretations we know that for every M, and in particular, for every M that models P: $M, \langle (D_1, E_1)\rangle \models \phi_1$.
Since $P, \langle (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$ then we also know $M, \langle (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ for every M that models P. By the serial conjunction case we can conclude that
$M, \langle (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$ for every M that models P. Consequently, as expected, it holds that $P, \langle (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$

r.3 Assume that $\mathcal{O}^t(D_0, D_1) \models \phi_1$ and $P, \langle (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ holds. By definition we know that for every M (and in particular, for every M that models P):
$M, \langle (D_0, E_1) \,{}^{\phi_1}\!\!\rightarrow (D_1, E_1)\rangle \models \phi_1$. Since $P, \langle (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$ then $M, \langle (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ for every M that models P. By the serial conjunction case we can conclude that $M, \langle (D_0, E_1) \,{}^{\phi_1}\!\!\rightarrow (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$. Consequently, as intended, it holds that $P, \langle (D_0, E_1) \,{}^{\phi_1}\!\!\rightarrow (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$

r.4 Assume that $\mathcal{O}^e(E_0, E_1) \models \phi_1$ and $P, \langle (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ holds. By definition we know that for every M (and in particular, for every M that models P):
$M, \langle (D_1, E_0) \,{}^{\phi_1}\!\!\rightarrow (D_1, E_1)\rangle \models \phi_1$. Since we know $P, \langle (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$ then $M, \langle (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ for every M that models P. By the serial conjunction case we can conclude that $M, \langle (D_1, E_0) \,{}^{\phi_1}\!\!\rightarrow (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$. Consequently, as intended, it holds that $P, \langle (D_1, E_0) \,{}^{\phi_1}\!\!\rightarrow (D_1, E_1) \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow (D_i, E_i)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$

r.5 Assume the following:

(a) There is an action-failed derivation starting in $\langle S_1\rangle, S_1 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$ ending in $\langle S_1 \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{j-1}}\!\!\rightarrow S_j\rangle, S_j \Vdash_P \psi$, for some serial-goal ψ

(b) $S_1 \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{p-1}}\!\!\rightarrow S_p$ is the rollback path of $S_1 \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{j-1}}\!\!\rightarrow S_j$ (cf. Definition 15)

(c) $\mathtt{Inv}(\mathtt{Seq}(\langle S_1 \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{p-1}}\!\!\rightarrow S_p\rangle)) = A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ (cf. Definition 16)

(d) $P, \langle S_p \,{}^{A_k^{-1}}\!\!\rightarrow \ldots {}^{A_1^{-1}}\!\!\rightarrow S_q\rangle, S_q \vdash_c A_k^{-1} \otimes \ldots \otimes A_1^{-1}$

and $P, \langle S_q \,{}^{A_q}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow S_i\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$.
We know that for every model M of P that $M, \langle S_q \,{}^{A_q}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow S_i\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$. If $\phi_1 \otimes \ldots \otimes \phi_k$ is completely unfolded, then the pre-conditions for the application of Lemma 7 are all verified. Otherwise, there is a complete unfolding, $Goal$, of $\phi_1 \otimes \ldots \otimes \phi_k$ iff $Goal$ can be obtained by a finite number of applications of rule 1 (something that follows directly from the definition of unfolding). In this case also, all pre-conditions for the application of Lemma 7 are verified. Thus, in both cases, there is a serial goal $\phi'_1 \otimes \ldots \otimes \phi'_{k'}$ (either the original one, or the unfolded $Goal$) for which we can apply Lemma 7, and $M, \langle S_q \,{}^{A_q}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow S_i\rangle \models \phi'_1 \otimes \ldots \otimes \phi'_{k'}$.
Applying Lemma 7 it follows that for every model M of P: $M, \langle S_1 \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{p-1}}\!\!\rightarrow S_p \,{}^{A_{p-1}^{-1}}\!\!\rightarrow \ldots {}^{A_1^{-1}}\!\!\rightarrow S_q\rangle \rightsquigarrow \phi'_1 \otimes \ldots \otimes \phi'_{k'}$, and since $M, \langle S_q \,{}^{A_q}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow S_i\rangle \models \phi'_1 \otimes \ldots \otimes \phi'_{k'}$, we can also conclude that: $M, \langle S_1 \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{p-1}}\!\!\rightarrow S_p \,{}^{A_{p-1}^{-1}}\!\!\rightarrow \ldots {}^{A_1^{-1}}\!\!\rightarrow S_q \,{}^{A_q}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow S_i\rangle \models \phi'_1 \otimes \ldots \otimes \phi'_{k'}$. Since rule 1. was already proven sound, then we can conclude that $M, \langle S_1 \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{p-1}}\!\!\rightarrow S_p \,{}^{A_{p-1}^{-1}}\!\!\rightarrow \ldots {}^{A_1^{-1}}\!\!\rightarrow S_q \,{}^{A_q}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow S_i\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$
Finally, by definition of the executional entailment it holds: $P, \langle S_1 \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_{p-1}}\!\!\rightarrow S_p \,{}^{A_{p-1}^{-1}}\!\!\rightarrow \ldots {}^{A_1^{-1}}\!\!\rightarrow S_q \,{}^{A_q}\!\!\rightarrow \ldots {}^{A_{i-1}}\!\!\rightarrow S_i\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$.

□

B.2.2. Completeness

We now show the auxiliary results needed for the proof of completeness of the $\vdash$ procedure.

We start by constructing a canonical model of the program P to link the model theory and the proof theory. We continue by proving that there are classical derivations for all serial conjunctions satisfied by that model, then consider the case of compensations, and finally prove that every atom or serial conjunction satisfied (with the general satisfaction relation) by the canonical model is obtained by the procedure. We end this section by proving that the canonical model is indeed a model.

Definition 31 (Canonical Model)**.** *The canonical model of a program P is the interpretation defined as follows:*

- $M_P(\pi) = \{a \in (\mathcal{L}_P \cup \mathcal{L}_\mathcal{O}) \mid P, \pi \vdash_c a\}$
- $M_P, \langle S_0 \,{}^{A_1}\!\!\rightarrow \ldots {}^{A_p}\!\!\rightarrow S_p\rangle \rightsquigarrow a$ *and a is an atom* ***only if****: $a \in \mathcal{L}_P$ and rule $\mathbf{r_5}$. of the Definition 22 is applicable to the resolvent $\langle S_0\rangle, S_0 \Vdash_P$*

a, resulting in the resolvent $\langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_p}{\rightarrow} S_p\rangle, S_p \Vdash_P a$

Remark 2. *Let* ϕ *and* ψ *be serial-Horn formulas:*

1. *If* $P, \pi_1 \vdash_c \phi$*,* $P, \pi_2 \vdash_c \psi$*, and* $\pi_1 \circ \pi_2$ *a split of* π *then* $P, \pi \vdash_c \phi \otimes \psi$
2. *If* $P, \pi_1 \vdash \phi$*,* $P, \pi_2 \vdash \psi$*, and* $\pi_1 \circ \pi_2$ *a split of* π *then* $P, \pi \vdash \phi \otimes \psi$
3. *If rule* $\mathbf{r_5}$*. of the Definition 22 is applicable to the resolvent* $\langle S_0\rangle, S_0 \Vdash_P \phi$ *resulting in the resolvent* $\langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_p}{\rightarrow} S_p\rangle, S_p \Vdash_P \phi$ *and* $P, \langle S_p \overset{A_p+1}{\rightarrow} \ldots \overset{A_{n-1}}{\rightarrow} S_n\rangle \vdash \phi$ *then* $P, \langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_p}{\rightarrow} S_p \overset{A_p+1}{\rightarrow} \ldots \overset{A_{n-1}}{\rightarrow} S_n\rangle \vdash \phi$
4. *If* $P, \pi \vdash \phi$ *then either* $P, \pi \vdash_c \phi$ *or there is a split* $\pi_1 \circ \pi_2$ *of* π *s.t.* $\pi_1 = \langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_p}{\rightarrow} S_p\rangle$*, there is a derivation starting in the resolvent* $\langle S_0\rangle, S_0 \Vdash_P \phi$ *resulting in the resolvent* $\langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_p}{\rightarrow} S_p\rangle, S_p \Vdash_P \phi$ *where rule* $\mathbf{r_5}$*. of the Definition 22 was applicable and* $P, \pi_2 \vdash \phi$*.*

Lemma 10. *If* $M_P, \pi \models_c \phi_1 \otimes \ldots \otimes \phi_k$ *then* $P, \pi \vdash_c \phi_1 \otimes \ldots \otimes \phi_k$

Proof. We prove by induction on the size of the formula $\phi_1 \otimes \ldots \otimes \phi_k$.
Base Case 1 (k = 0): $M_P, \pi \models_c ()$ iff π is a 1-path $\langle S\rangle$. If this is the case, then the derivation starting in the resolvent $\langle S\rangle, S \Vdash_P ()$ succeeds, for every state S. Since we have not used rule $\mathbf{r_5}$. for this derivation, we can conclude $P, \pi \vdash_c ()$
Base Case 2 (k = 1): $M_P, \pi \models_c \phi$ and ϕ is an atom, then by definition of M_P we know that $P, \pi \vdash_c \phi$
Induction Step (k = j+1): Suppose the hypothesis is true for $\phi_1 \otimes \ldots \otimes \phi_j$.

$M_P, \pi \models_c \phi_1 \otimes \ldots \otimes \phi_{j+1}$ implies that there is a split $\pi_1 \circ \pi_2$ of π s.t. $M_P, \pi_1 \models_c \phi_1 \otimes \ldots \otimes \phi_j$ and $M_P, \pi_2 \models_c \phi_{j+1}$. Applying the Induction Hypothesis to π_1 we know that $P, \pi_1 \vdash_c \phi_1 \otimes \ldots \otimes \phi_j$. Since ϕ_{j+1} is an atom, from Base Case 2 follows $P, \pi_2 \vdash_c \phi_{j+1}$.

Since $P, \pi_1 \vdash_c \phi_1 \otimes \ldots \otimes \phi_j$ and $P, \pi_2 \vdash_c \phi_{j+1}$ we conclude $P, \pi \vdash_c \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ □

Lemma 11. *If* $M_P, \langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_p}{\rightarrow} S_p\rangle \rightsquigarrow \phi_1 \otimes \ldots \otimes \phi_k$ *then rule* $\mathbf{r_5}$*. of the Definition 22 is applicable to the resolvent* $\langle S_0\rangle, S_0 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$ *resulting in the resolvent* $\langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_p}{\rightarrow} S_p\rangle, S_p \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$

Proof. We prove by induction on the size of the formula $\phi_1 \otimes \ldots \otimes \phi_k$. Note that $M, \pi \rightsquigarrow ()$ is impossible for every path π.
Base Case (k = 1): $M_P, \langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_p}{\rightarrow} S_p\rangle \rightsquigarrow \phi$ and $\phi \in \mathcal{L}_P$ (as $\phi \in \mathcal{L}_\mathcal{O}$ then $M, \pi \rightsquigarrow \phi$ is impossible for every path π).

Then by definition of M_P we know that rule $\mathbf{r_5}$. of the Definition 22 is applicable to the resolvent $\langle S_0\rangle, S_0 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$ resulting in the resolvent $\langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_p}{\rightarrow} S_p\rangle, S_p \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$
Induction Step (k = j+1): Suppose the hypothesis is true for $\phi_1 \otimes \ldots \otimes \phi_j$.

$M_P, \pi \rightsquigarrow \phi_1 \otimes \ldots \otimes \phi_{j+1}$. In this case we have two possible scenarios:

1. $M_P, \pi \rightsquigarrow \phi_1 \otimes \ldots \otimes \phi_j$ and thus by Induction Hypothesis we know that rule $\mathbf{r_5}$. of the Definition 22 is applicable to the resolvent $\langle S_0\rangle, S_0 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_j$ resulting in the resolvent $\langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_p}{\rightarrow} S_p\rangle, S_p \Vdash_P \phi_1 \otimes \ldots \otimes \phi_j$. If this is the case, by definition of rule $\mathbf{r_5}$. we know that it is also applicable to the resolvent $\langle S_0\rangle, S_0 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ resulting in the resolvent $\langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_p}{\rightarrow} S_p\rangle, S_0 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$, for a path $\pi = \langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_p}{\rightarrow} S_p\rangle$.
2. $M_P, \pi \not\rightsquigarrow \phi_1 \otimes \ldots \otimes \phi_j$ and $M_P, \pi \rightsquigarrow \phi_1 \otimes \ldots \otimes \phi_{j+1}$. In this case, we know that there exist a path π_1, π_0, π_r s.t. $M_P, \langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_j}{\rightarrow} S_j\rangle \models_c \phi_1 \otimes \ldots \otimes \phi_j$ and $M_P, \langle S_j\rangle \models_p \phi_{j+1}$ $M_P, \langle S_j\rangle \not\models_c \phi_{j+1}$, π_0 a the rollback path of π_1 and, a path $M_P, \pi_r \models_c \mathtt{Inv}(\mathtt{Seq}(\pi_0))$ and $\pi = \pi_0 \circ \pi_r$. If this is the case, by Lemma 10 there must exist a classical derivation $P, \langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_j}{\rightarrow} S_j\rangle \vdash_c \phi_1 \otimes \ldots \otimes \phi_j$. By definition of a classical action-failed derivation we also know that there is a classical action-failed derivation starting in $\langle S_j\rangle, S_j \Vdash_P \phi_{j+1}$ and thus there is a classical action-failed derivation starting in $\langle S_0\rangle, S_0 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ and ending in $\langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_j}{\rightarrow} S_j\rangle, S_j \Vdash_P \phi_{j+1}$.
   Since, by Lemma 10 it follows that $P, \pi_r \vdash_c \mathtt{Inv}(\mathtt{Seq}(\pi_0))$, then all conditions of rule $\mathbf{r_5}$. of the Definition 22 hold, and so we can apply it to the resolvent $\langle S_0\rangle, S_0 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ resulting in the resolvent $\langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_p}{\rightarrow} S_p\rangle, S_p \Vdash_P \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$, where $\langle S_0 \overset{A_1}{\rightarrow} \ldots \overset{A_p}{\rightarrow} S_p\rangle = \pi_0 \circ \pi_r$.

□

Lemma 12. *Let* π *be a 1-path of the form* $\pi = \langle S\rangle$*. If* $M_P, \pi \models \phi_1 \otimes \ldots \otimes \phi_j$ *then* $P, \pi \vdash \phi_1 \otimes \ldots \otimes \phi_j$

Proof. We apply induction on the size of the serial-Horn formula $\phi_1 \otimes \ldots \otimes \phi_j$:

Base Case 1 (j = 0):
$M_P, \pi \models ()$. If this is the case, then there is a derivation starting in the resolvent $\langle S \rangle, S \Vdash_P ()$ and successfully ending in the resolvent $\langle S \rangle, S \Vdash_P ()$. And thus both $P, \pi \vdash_c ()$ and $P, \pi \vdash ()$ are true.
Base Case 2 (j = 1):
$M_P, \pi \models \phi$ and ϕ is an atom. If this is the case, and since $M_P, \pi \rightsquigarrow \phi$ is not defined for paths smaller than size 2, then $M_P, \pi \models \phi$ iff $\phi \in M_P(\pi)$. Moreover, by definition of M_P this implies that $P, \pi \vdash_c \phi$ which implies $P, \pi \vdash \phi$.
Inductive Case (j = k+1):
Suppose the hypothesis holds for values up to k.

Since π is a 1-path and $M_P, \pi \rightsquigarrow \phi_1 \otimes \ldots \otimes \phi_k \otimes \phi_{k+1}$ is not defined for paths smaller than size 2, then $M_P, \pi \models \phi_1 \otimes \ldots \otimes \phi_k \otimes \phi_{k+1}$ iff there are $k+1$ splits of π s.t. $M, \pi_i \models \phi_i$ $(1 \leq i \leq k+1)$.

However, since π is a 1-path, the only possible split to make is $\pi_i = \pi$. Thus for $M_P, \pi \models \phi_1 \otimes \ldots \otimes \phi_k \otimes \phi_{k+1}$ to hold, $M, \pi \models \phi_i$ must hold for all formulas ϕ_i in $\phi_1 \otimes \ldots \otimes \phi_k \otimes \phi_{k+1}$. If this is the case, by Base Case 2, for every formula, $P, \pi \vdash \phi_i$, and thus $P, \pi \vdash \phi_1 \otimes \ldots \otimes \phi_k \otimes \phi_{k+1}$ □

Lemma 13. *Let a be an atom. If $M_P, \pi \models a$ then $P, \pi \vdash a$*

Proof. We apply induction on the size of the k-path π.
Base Case (k = 1):
$M_P, \pi \models a$ iff $a \in M_P(\pi)$ and thus $P, \pi \vdash_c atom$ and $P, \pi \vdash a$
Induction step (k = m+1):
$M_P, \pi \models a$ if:

- $a \in M_P(\pi)$ and thus $P, \pi \vdash_c atom$ and also $P, \pi \vdash a$; or
- $M_P, \pi_1 \rightsquigarrow a$ and $M_P, \pi_2 \models a$. If this is the case, then π_2 must be strictly smaller than π, as π_1 must be at least a 2-path. So, by Induction Hypothesis, we can conclude that $P, \pi_2 \vdash a$.
  Moreover, by definition of M_P, we know that $M_P, \pi_1 \rightsquigarrow a$ implies that there is a derivation starting in the resolvent $\langle S_0 \rangle, S_0 \Vdash_P a$ where rule $\mathbf{r_5}$. of the Definition 22 is applicable and resulting in the resolvent $\langle S_0 \xrightarrow{A_1} \ldots \xrightarrow{A_p} S_p \rangle, S_p \Vdash_P a$. In this case $\pi_1 = \langle S_0 \xrightarrow{A_1} \ldots \xrightarrow{A_p} S_p \rangle$. From this, we conclude $P, \pi \vdash a$.

□

Lemma 14. *If $M_P, \pi \models \phi_1 \otimes \ldots \otimes \phi_j$ then $P, \pi \vdash \phi_1 \otimes \ldots \otimes \phi_j$*

Proof. We prove by lexicographic induction first on the size of the k-path π and then the size of the serial-Horn formula $\phi_1 \otimes \ldots \otimes \phi_j$
Base Case (k = 1):
π is a 1-path and $M_P, \pi \models \phi_1 \otimes \ldots \otimes \phi_j$. In this case we can apply Lemma 12 and conclude $P, \pi \vdash \phi_1 \otimes \ldots \otimes \phi_j$.
Induction step (k = m+1):
Assume the hypothesis holds for paths with size up to m.

$M_P, \pi \models \phi_1 \otimes \ldots \otimes \phi_j$ holds if one the two cases is true:

- *Serial Conjunction Case:* There is a split $\pi_1 \circ \pi_2$ of π s.t. $M, \pi_1 \models \phi_1$ and $M, \pi_2 \models \phi_2 \otimes \ldots \otimes \phi_j$. In this case we know that either π_1 or π_2 are strictly smaller than π and both ϕ_1 and $\phi_2 \otimes \ldots \otimes \phi_j$ are strictly smaller than $\phi_1 \otimes \ldots \otimes \phi_j$. As such, and since we know by Lemma 13 that for an atomic formula ϕ $M_P, \pi \models \phi$ then $P, \pi \vdash \phi$. Then by Induction Hypothesis we have $P, \pi_1 \vdash \phi_1$ and $P, \pi_2 \vdash \phi_2 \otimes \ldots \otimes \phi_j$. From this we know $P, \pi \vdash \phi_1 \otimes \ldots \otimes \phi_j$
- *Compensating Case:* There is a split $\pi_1 \circ \pi_2$ of π s.t. $M, \pi_1 \rightsquigarrow \phi_1 \otimes \ldots \otimes \phi_j$ and $M, \pi_2 \models \phi_1 \otimes \ldots \otimes \phi_j$. Since π_1 must be at least a 2-path, then we know that both π_1 and π_2 are strictly smaller than π. As such we can apply the Induction Hypothesis to π_2 and conclude $P, \pi_2 \vdash \phi_1 \otimes \ldots \otimes \phi_j$. Moreover, since we know $M, \pi_1 \rightsquigarrow \phi_1 \otimes \ldots \otimes \phi_j$ then by Lemma 11 we know that rule $\mathbf{r_5}$. of the Definition 22 is applicable to the resolvent $\langle S_0 \rangle, S_0 \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$ resulting in the resolvent $\langle S_0 \xrightarrow{A_1} \ldots \xrightarrow{A_p} S_p \rangle, S_p \Vdash_P \phi_1 \otimes \ldots \otimes \phi_k$, for $\pi_1 = \langle S_0 \xrightarrow{A_1} \ldots \xrightarrow{A_p} S_p \rangle$. Consequently, since $\pi_1 \circ \pi_2$ are splits of π, it follows that $P, \pi \vdash \phi_1 \otimes \ldots \otimes \phi_j$

□

Lemma 15. *Let a be an atom.*
If $P, \pi \vdash a$ then $M_P, \pi \models a$

Proof. We prove by induction on the size of π.
Base Case (k = 1):
If $P, \pi \vdash a$ then either a is an oracle primitive and $\mathcal{O}^d(\pi) \models a$ (i.e. rule $\mathbf{r_2}$.) , or we can apply rule $\mathbf{r_1}$. together with rule $\mathbf{r_2}$. an arbitrary number of times to reach the resolvent $\pi, S \Vdash_P ()$. Note that rule $\mathbf{r_5}$. is never applicable in this case.

As such, if π is a 1-path $P, \pi \vdash a$ implies $P, \pi \vdash_c a$ and thus $a \in M(\pi)$ and $M, \pi \models a$

Inductive step (k = m+1):
Assume this holds for paths of size up to m.

If $P, \pi \vdash a$ then one of the two cases is true:

1. $P, \pi \vdash_c a$ and in this case by definition of M_P we know that $a \in M_P(\pi)$ and thus $M_P, \pi \models a$
2. There is a split $\pi_1 \circ \pi_2$ of π s.t. $\pi_1 = \langle S_0 \, {}^{A_1}\!\!\rightarrow \ldots {}^{A_p}\!\!\rightarrow S_p \rangle$, where the resolvent $\langle S_0 \rangle, S_0 \Vdash_P \phi$ results in the resolvent $\langle S_0 \, {}^{A_1}\!\!\rightarrow \ldots {}^{A_p}\!\!\rightarrow S_p \rangle, S_p \Vdash_P \phi$ by applying rule $\mathbf{r_5}$. of the Definition 22 and $P, \pi_2 \vdash \phi$. Since π_2 is strictly smaller than π, then by Induction Hypothesis we know that $M, \pi_2 \vdash \phi$. Moreover by definition of M_P, since there is a resolvent $\langle S_0 \rangle, S_0 \Vdash_P \phi$ which results in the resolvent $\langle S_0 \, {}^{A_1}\!\!\rightarrow \ldots {}^{A_p}\!\!\rightarrow S_p \rangle, S_p \Vdash_P \phi$ by applying rule $\mathbf{r_5}$. of the Definition 22, it follows that $M_P, \pi_1 \rightsquigarrow a$. Consequently, by the compensating case of $\models$, it follows that $M_P, \pi \models a$.

□

Lemma 16. *M_P is a model of P.*

Proof. For M_P to be a model of P we have to prove for every rule $h \leftarrow body$:

(1) if $M_P, \pi \models body$ then $M_P, \pi \models head$
(2) if $M_P, \pi \models_c body$ then $M_P, \pi \models_c head$
(3) if $M_P, \pi \rightsquigarrow body$ then $M_P, \pi \rightsquigarrow head$

We prove each claim in turn:

(1) Assume $M_P, \pi \models body$, then by Lemma 14 we know that $P, \pi \vdash body$. By definition of $\vdash$, since the rule $h \leftarrow body$ exists in P, this is equivalent to $P, \pi \vdash head$. Then, by Lemma 15, $M_P, \pi \models head$.
(2) Assume $M_P, \pi \models_c body$, then by Lemma 10 we know that $P, \pi \vdash_c body$. By definition of $\vdash$, since the rule $h \leftarrow body$ exists in P, this is equivalent to $P, \pi \vdash_c head$. Then by definition of M_P we know that $M_P, \pi \models_c head$.
(3) Assume now that $M_P, \pi \rightsquigarrow body$. Then by Lemma 11 we know that, for $\pi = \langle S_0 \, {}^{A_1}\!\!\rightarrow \ldots {}^{A_p}\!\!\rightarrow S_p \rangle$, rule $\mathbf{r_5}$. of the Definition 22 is applicable to the resolvent $\langle S_0 \rangle, S_0 \Vdash_P body$ resulting in the resolvent $\langle S_0 \, {}^{A_1}\!\!\rightarrow \ldots {}^{A_p}\!\!\rightarrow S_p \rangle, S_p \Vdash_P body$.
From this we can apply rule $\mathbf{r_1}$. and conclude $\langle S_0 \, {}^{A_1}\!\!\rightarrow \ldots {}^{A_p}\!\!\rightarrow S_p \rangle, S_p \Vdash_P head$. If this is the case, then we could have started in the resolvent $\langle S_0 \rangle, S_0 \Vdash_P head$ and apply $\mathbf{r_5}$. resulting in the resolvent $\langle S_0 \, {}^{A_1}\!\!\rightarrow \ldots {}^{A_p}\!\!\rightarrow S_p \rangle, S_p \Vdash_P head$ (as rule $\mathbf{r_1}$. could be performed as part of the classical action-failed derivation in the first step of rule $\mathbf{r_5}$.).
As such, there is a derivation starting in $\langle S_0 \rangle, S_0 \Vdash_P head$ resulting in the resolvent $\langle S_0 \, {}^{A_1}\!\!\rightarrow \ldots {}^{A_p}\!\!\rightarrow S_p \rangle, S_p \Vdash_P head$. Consequently, by definition of M_P, $M_P, \pi \rightsquigarrow head$.

□