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 ACID model of transactions. Contrarily, transactional properties over actions executed externally need to be relaxed, as it is impossible to rollback 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 deal with the problem of evolving its knowledge base (KB) by means of updates and actions.

Independently of the semantics chosen for a given application, it is often important to guarantee properties on the execution of these updates as for instance, that the KB is always left consistent independently on what and how updates are executed. The problem of what properties should be ensured has been widely studied in the database community, where actions are required to follow the ACID transactional model in order to be considered reliable. ACID stands for Atomicity, Consistency, Isolation and Durability, and an ACID transaction is a set of actions that must be executed respecting all of these properties.

Transaction Logic ($\mathcal{TR}$) is an extension of predicate logic proposed in [6] to reason and execute updates following this ACID transaction model *independently*

[1]Some of the results in this paper were preliminary presented in [25] and [26]. This work was partially supported by project ERRO (PTDC/EIA-CCO/121823/2010).
[*]Author supported by the FCT grant SFRH/BD/64038/2009

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 with the semantics, can answer practical questions like "can this action be executed in this state" or "how does my database evolve if this action 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 state change semantics 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 [49], argumentation theories [18], AI planning [6], workflow management and semantic web services [50], databases [9], and general KB representation [5].

Example 1 ($\mathcal{TR}$ Financial Transactions). *As 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; $deposit(Amt, Acnt)$ 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 $\otimes$ denotes serial conjunction. Thus, $withdraw(Amt, Acnt) \otimes deposit(Amt, Acnt')$ means execute the (trans)action $withdraw(Amt, Acnt)$ and immediately afterwards execute $deposit(Amt, Acnt')$.*

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

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.

A key feature of $\mathcal{TR}$ is that, unlike many other logic systems, the KB is imposed to evolve only into consistent states respecting ACID properties as it is required in databases. As a result, instead of providing means to reason about what formulas are true in a KB (as e.g. [22,1,21]), or about the direct and indirect effects resulting from a given action in a knowledge base (as e.g. [23,34,40,54]), the semantics of $\mathcal{TR}$ talks about *how* an update can be executed in a KB respecting the ACID model. That is, given a fixed semantics of states and a fixed semantics of updates (given as a parameter, by oracle instantiations), $\mathcal{TR}$ semantics specifies what are the paths that allow an update to succeed following the ACID model. Then, $\mathcal{TR}$ statements have the form $P, (D_1, D_2, \ldots, D_{n-1}, D_n) \models t$ with the meaning that transaction t succeeds in program P when executed in the (arbitrary) state D_1 by changing the system into state D_n through the path $D_1, D_2, \ldots, D_{n-1}, D_n$.

As a result of this abstraction, $\mathcal{TR}$ is especially suited for dealing with transactions in contexts where the semantics is not fixed a priori, or where one has to deal with KBs each equipped with a different semantic, such as the Semantic Web.

However, as an inherent consequence of forcing every transaction to be ACID, $\mathcal{TR}$ fails to model situations where some ACID properties need to be relaxed. Since every action is interpreted as a strict ACID transaction, $\mathcal{TR}$ requires domains where a complete control and specification of the KB exists. As a result, $\mathcal{TR}$ is not able to express richer situations where the internal database needs to interact and execute actions in external domains, as e.g. an interaction with a web-service, with several ontologies in the web, or with a real external entity. The problem about external domains is that it is no longer possible to ensure the same ACID transaction model as in the internal KB. In fact, since one does not control the external environment on which these actions are executed, the rollback of external actions is impossible. However, in case of a transaction failure, something must be done to preserve some kind of external consistency if an external action was previously performed.

Example 2 (Travel)**.** *As a very simple example, consider a transaction that makes a reservation for a travel in some given dates D, comprising both the booking of a hotel room and flight. Such a transaction can be written (in a very simplified way) in a $\mathcal{TR}$-like form as follows:*

$$\begin{aligned}&bookTravel(City, D) \leftarrow\\ &\quad findHotel(City, D, H) \otimes findFlight(City, D, F)\\ &\quad \otimes reserveHotel(H, D) \otimes reserveFlight(F, D)\end{aligned}$$

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.

If such a transaction fails e.g. because the flight is no longer available, then something must be done about the previously reserved hotel. But since the KB taking care of hotel reservations is external, simply rolling back might not 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 describing knowledge about an organization, storing information about customers, sales, etc. As in a database, this KB should always comply with the ACID properties. I.e. if some action performed by the organization fails, then its internal knowledge base should be rolled back to a consistent state.*

In a general setting, 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 custom'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:

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

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, complying with the ACID properties means that, if the dispatch action fails, then 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:

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

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 really 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.

As an example requiring more elaborate KBs, that is explored further in the remainder of this paper:

Example 4 (Diagnosis example)**.** *Consider 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 a Description Logic 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 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 the need to ensure transaction properties on environments that have both an internal and an external component. In this sense, the idea of what to do to impose external consistency corresponds with the notion of *compensation*, proposed originally in the database literature for long-running transactions [20]. Whenever rollbacking 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 consistency and atomicity externally.

In this work we propose $\mathcal{ETR}$, an extension of $\mathcal{TR}$ to reason and execute transactions executed on knowledge bases defined by an internal KB and an external KB. While actions performed in the internal KB follow the strict ACID model, actions executed externally follow a relaxed model based on compensations.

As in $\mathcal{TR}$, assuming this external oracle allows $\mathcal{ETR}$ to reason and execute transactions that require interaction with external sources without committing to any semantics for the external KB. By instantiating this external oracle with a Description Logics [1] semantics, or with logics for dynamic external domains like Action Languages [23] or Event Calculus [34], $\mathcal{ETR}$ 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 no external actions are presented. Afterwards we construct a 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 Logic oracles for both the internal and external KB, and the dynamic description languages – Action Languages, Situation Calculus and Event Calculus – for describing 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 the enunciated 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 a 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 $\phi \wedge \psi$ defines the non-deterministic 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 to 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 as neither is the definition of a state. Consequently, to distinguish between states, $\mathcal{TR}$ works with a set of state identifiers that uniquely identify a state. Then $\mathcal{O}^d$ 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 consider to be true in d. 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 denoted as elementary transitions.

The data and transition oracle are strongly related. Particularly, the state identifiers of these two oracles are defined under 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, states 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 a disk page or a XML file. Since a state identifier uniquely identifies a state, from this moment forward we employ the terms of "state" and "state identifier" interchangeably.

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

2.1. $\mathcal{TR}$ Model Theory

Since the goal of $\mathcal{TR}$ is to impose transactional properties on abstract knowledge 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 respecting the ACID properties. As a result, contrarily 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 a 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 for the internal and external KB which all interpretations must model. By limiting the set of possible interpretations to satisfy these restrictions, we can force 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* $\mathcal{O}^t(D_1, D_2) \models \varphi$

Afterwards, satisfaction of complex formulas over paths, requires the prior definition of 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 $\pi = \pi_1 \circ \pi_2$.*

[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.

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

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 $b.ins \in M(\{a\}, \{a, b\})$. Then, it holds $M, \langle\{\}, \{a\}\rangle \models a.ins$ and $M, \langle\{\}, \{a\}, \{a, b\}\rangle \models a.ins \otimes b.ins$.*

2.2. $\mathcal{TR}$ Logical Entailment

After defining how satisfaction of complex formulas is performed w.r.t. a given interpretation, we now define which of these interpretations model a formula and a program.

Definition 4 (Models)**.** *An interpretation M is a model of an $\mathcal{TR}$ 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 is an atom in the language and ϕ is any complex formula. Since rules are just complex formulas, a program can be seen as a set of formulas. To say that M models a given program implies that M models every rule in every path. Intuitively and as intended, this means that in every path π, 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 in an atomic way. Non-deterministic transactions are possible, in which 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 $S_1, \ldots, S_n$ be a path:*

$$P, (S_1, \ldots, S_n) \models \phi \qquad (1)$$

is true 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 $S_1 \ldots, S_n$ that makes (1) true.

Example 7. *Recall example 6. Here $P, (\{\}, \{a\}) \models p$, $P, (\{a\}, \{a, b\}) \models q$ and $P, (\{\}, \{a\}, \{a, b\}) \models t$. Also, $P, (\{a\}, \{a, c\}) \models q$ and $P, (\{\}, \{a\}, \{a, c\}) \models t$, making t and q non-deterministic transactions.*

Based on this, $\mathcal{TR}$ defines a proof theory and 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 $n \geq 0$.

For this serial-Horn version of $\mathcal{TR}$, a proof theory sound and complete with the executional entailment exists, providing the theory for *executing* $\mathcal{TR}$ programs in a top-down manner. 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 $D_0, D_1, \ldots, D_n$ that makes the formula $P, D_0, D_2, \ldots, D_n \models G$ true.

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 giving 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 tautology formula that is true in every path of length 1.*

Then, 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 and be an atomic formula, while ϕ and $rest$ are 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 say $P, D_0, D_1, \ldots, D_n \vdash G$ 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 $P, D_0, D_1, \ldots, D_n \vdash G$ iff $P, D_0, D_1, \ldots, D_n \models G$*

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 $external\_b$ succeeds; or if the insertion of predicate q followed by $external\_c$ succeeds.

However, let's consider 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 rollback 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), rollback $p.ins$ 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 were 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.

This is as expected 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 try. Thus, in a implementation perspective, whenever the system makes a choice that leads to a non-successful derivation, then it simply rollbacks the state previous to that choice and tries to succeed in an alternative branching. In a 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 is no longer the case. Since it is impossible to rollback an external state, then the alternative is to compensate for the external actions already executed, and then succeed in an alternative branching. Contrarily 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 previous two 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 form this failure in π_1 (by rollbacking internally and compensating externally) and succeeds in π_2.

The first two satisfaction relations represent the building blocks for defining failures and are *not* use 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 rollback internally and compensate externally. This is encoded in $M, \pi \leadsto \phi$ 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 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.

Next we continue by defining $\mathcal{ETR}$'s theory. 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 with $\mathcal{ETR}$'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 to 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 and trivial success in the external domain.

Based on the latter notion of language, $\mathcal{ETR}$ formulas are defined as follows.

Definition 8 ($\mathcal{ETR}$ Atoms, Formulas and Programs)**.** *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 $\mathtt{ext}(a, b_1 \otimes \ldots \otimes b_j)$ 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. 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.

One of the novelties in $\mathcal{ETR}$'s semantics is the retrieval of some paths were a formulas 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 precisely deal with this, we slightly restrict the language of $\mathcal{ETR}$ w.r.t. negation. This is a technical detail that will allow us to better handle failures of formulas. As a result, negation is only applicable to atoms, and thus $\phi \leftarrow \psi$ can no longer be syntactic sugar for $\phi \vee \neg\psi$ (since ψ is defined as a complex formula).

External actions can appear in a program in two different ways: 1) without any kind of compensation associated, i.e. $\mathbf{ext}(a, \mathtt{nop})$, and in this case we write $\mathbf{ext}(a)$ 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 $a, b_i \in \mathcal{L}_a$. Note that there is no explicit relation between a and $b_1 \otimes \ldots \otimes b_j$, and that it is possible to define different compensating actions for the same action a in the same program. It is thus the programmer's task to determine which is the correct compensation for action a in a given moment for a given rule.

Based on this, 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 $a, b_i \in \mathcal{L}_a$

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 eligible 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 and update this DL, the agent has the primitives $\mathtt{dlquery}()$ *and* $\mathtt{dladd}()$. *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:*

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

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 seen 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 battle 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 ac-

tions 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:*

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

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 external KB semantics is left opened. If we assume nothing on the semantics of states and updates, then it is also impossible to reason on how to repair the effects of a particular action in a given KB.

However, if such assumptions are made regarding the semantics of the external environment, 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 [26] and we refer the interested reader to such work for additional details.

Next 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 reason about the semantics 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 and external KB, two disjoint sets of state identifiers are needed: one for internal states, and another for uniquely identifying states of the external domain (*external states*). Then, $\mathcal{O}^e$ is a mapping from a pair of external states identifiers into 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 from state identifier E_1 into state identifier E_2.

Dealing with state identifiers instead of materialized states is of particular importance when considering external domains. In fact, it may be at all 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. Then, 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 following we will use the the terms state as state id.

Since we stipulated that the actions `failop` and `nop` always belong to $\mathcal{L}_a^*$ with a precise meaning, we also force 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. $\mathcal{O}^e(E_1, E_2) \models \mathbf{ext}(a, b_1 \otimes \ldots \otimes b_j)$ 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. Thus, we assume that every external primitive can cause a state transition and define $\mathcal{O}^e$ as a function that maps primitives to a pair of states. 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 Theory we also elaborate upon the role of this external oracle in Section 4.2 and show how can be instantiated using 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. 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 just a sequence of states defined as follows.

Definition 9 (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, $S_1,^{A_1} \ldots ,^{A_{k-1}} S_k$ where A_is ($1 \leq i < k$) are atoms from $\mathcal{L}_a^*$ or $\mathcal{L}_i$.*

For convenience and to help the semantics recover from external failures, the previous definition of path also records the operations performed between states. Then $S_i,^{A_i}, S_{i+1}$ 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 allow us to know that $b_1 \otimes \ldots \otimes b_j$ is the compensation to be performed in the advent of a failure after the execution of the external action a.

Interpretations are then defined in the usual way, but now catering also for the external oracle.

Definition 10 (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 formula φ:*

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

Interpretations in $\mathcal{ETR}$ (and as referenced also the language of programs) are slightly restricted when compared to $\mathcal{TR}$. Since we are dealing with an oracle that is external, it now makes little sense to allow interpretations to satisfy external primitives in arbitrary paths. This precludes the possibility of redefining oracles primitives in a $\mathcal{ETR}$ program, making oracles primitives only true whenever the oracles say it so. Although we could have introduced this restriction solely on external primitives, we decided to impose the overall separation between oracle primitives and complex actions defined in a program. This makes the logic cleaner without particularly limiting the expressivity of the language. In fact, note that every $\mathcal{TR}$ program can be re-written to comply with this restriction and achieve the same results. For instance if the rule $a.ins \leftarrow c.ins \otimes d.ins$ exists in the program, then we can re-write it as:

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

and substitute all the remaining occurrences of $a.ins$ in the program by $new\_a.ins$.

Besides the notion of path splits inherited from $\mathcal{TR}$, we require additional notions on paths as the prefix and the ending of a path. These are intuitive notions that will be needed to isolate the exact point of failure of a transaction.

Definition 11 (Prefix of a Path). *A* prefix *of a k-path π corresponds to a m-path (where $1 \leq m \leq k$) obtained from π where the order of states and the corresponding annotated transitions is preserved.*

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

The definition of satisfaction of the standard $\mathcal{TR}$ can now easily be generalized to the notion of path in $\mathcal{ETR}$, where states are pairs with the internal and external state:

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.*

This latter satisfaction, coming from $\mathcal{TR}$, 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 some external 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.

Note that, since $\mathcal{TR}$ only deals with domains where one has complete control, this need to deal with failures in the perspective of $\mathcal{TR}$'s model theory does not exist. In fact, since in case of internal failures, a previous consistent state can just be reinstated, then $\mathcal{TR}$'s model theory only needs to obtain the paths where the formula completely succeeds. Contrarily, this not the case in $\mathcal{ETR}$. Since rollbacking of external actions is often impossible, $\mathcal{ETR}$'s model theory needs also 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. The idea is to, given a path, determine the formulas that either "completely" succeed, or at least succeed up to some point and then fail to execute some action or query.

Example 10 (Running Example)**.** *Recall example 8, but where 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).

Then, formulas $p.ins \otimes \mathbf{ext}(a, a_1 \otimes a_2)$ and $q.ins \otimes \mathbf{ext}(c, c_1)$ are classically satisfied respectively in paths $\langle(\{\}, E_1),^{p.ins} (\{p\}, E_1),^{\mathbf{ext}(a,a_1 \otimes a_2)} (\{p\}, E_2)\rangle$ and $\langle(\{\}, E_1),^{q.ins} (\{q\}, E_1),^{\mathbf{ext}(c,c_1)} (\{q\}, E_5)\rangle$. Furthermore, it is easy to check that $\mathbf{ext}(b, b_1)$ cannot succeed in any path starting in state E_2 (given the external oracle definition).

The idea of partial satisfaction is to identify the path $\langle(\{\}, E_1),^{p.ins} (\{p\}, E_1),^{\mathbf{ext}(a,a_1 \otimes a_2)} (\{p\}, E_2)\rangle$ 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.

In partial satisfaction, we allow primitives to fail at some point, or to completely succeed. In the case of the serial conjunction, when satisfying $\phi \otimes \psi$, if ϕ cannot succeed classically, it suffices to satisfy part of the conjunction. 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.* $M, \langle(D, E),^{\phi} (D_i, E)\rangle \models_c \phi$
   (c) $M, \pi \not\models_c \phi$, $\phi \in \mathcal{L}_a^*$, $\pi = \langle(D, E)\rangle$ *and* $\neg\exists E_i$ *s.t.* $M, \langle(D, E),^{\phi} (D, E_i)\rangle \models_c \phi$
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* $M, \pi_2 \models_p \psi$

For the usage below of this definition, it is crucial to know the exact failure point of the transaction, so that external actions priorly performed can be compensated. For that, the previously defined partial satisfaction statement $M, \pi \models_p \phi$ says that either ϕ succeeds in path π w.r.t. to the interpretation M, *or* ϕ "legitimately" fails in π (and in this case, the last state in π corresponds to the exact point of failure).

There are several reasons for a formula to not be satisfied in a given path. Obviously, given a relational oracle, the query a cannot be classically satisfied in the path $\langle(\{a\}, E),^{b.ins} (\{a, b\}, E)\rangle$ for any interpretation M. Similarly, the action $b.ins$ does not succeed 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 obtained by any of the singleton states $\langle(\{a\},E)\rangle$ and $\langle(\{a,b\},E)\rangle$. As a result, although for any M, $M,\langle(\{a\},E),^{b.ins}(\{a,b\},E)\rangle \not\models_c a$ (i.e. a fails), it is also the case that $M,\langle(\{a\},E),^{b.ins}(\{a,b\},E)\rangle \not\models_p a$.

So the partial satisfaction definition only deals with with failures that come from a real impediment of 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 (points 1b and 1c respectively).

Based on this, we define "legitimate" fails the ones where a formula is partially but not classically satisfied in a path. Moreover, the path where such happens must always end exactly in the state prior to the failure (cf. point 1 of Proposition 1 below), and it is why failures of primitives are constraint to 1-paths in points 1b and 1c. These 1-paths represent the state where the transaction failed.

Next in Proposition 1 we enunciate some additional properties of the partial satisfaction definition. In this sense, point 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 formulas without negation, the partial satisfaction is a relaxed version of the classical satisfaction, (point 3) and the two satisfaction relations coincide whenever they are evaluating atoms that are not specified by the oracles (point 4).

Proposition 1. *Let M be an interpretation, π a path, $\pi_{\mathtt{end}}$ the 1-path containing the last state of π, ϕ and ψ any $\mathcal{ETR}$ formulas, ϕ' a formula without negation, 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. $M,\pi_{\mathtt{end}}\models_p a$ and $M,\pi_{\mathtt{end}}\not\models_c a$*
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. Just like in logic programming, transaction t can be seen as the disjunction of the two bodies of the rules. In this sense, in $\mathcal{TR}$ and in logic programming we can satisfy t by satisfying either the first body or the second. Furthermore, in $\mathcal{ETR}$ we consider an additional way of satisfying such disjunction. Namely, the disjunction is satisfied if the first body is "tried", compensated, and then the second disjunct 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 need to be defined. We start by defining the notion of a *rollback path*. This path is the first step for recovering consistency of a KB with an internal and external component. Here one has to collect all external actions that have been executed in a path and need to be compensated; and to rollback the internal state. This is encoded as follows:

Definition 15 (Rollback Path, and Sequence of External Actions)**.** *Let π be a k-path of the following form $\langle(D_1,E_1),^{A_1}(D_2,E_2),^{A_2}\dots,^{A_{k-1}}(D_k,E_k)\rangle$. 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 $\mathtt{Seq}(\pi)$, 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.

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

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

Definition 16 (Inversion, and Recovery Path)**.** *Let $S=\langle\mathbf{ext}(A_1,\mathcal{A}_1^{-1}),\ldots\mathbf{ext}(A_n,\mathcal{A}_n^{-1})\rangle$ be a sequence of actions from $\mathcal{L}_a^*$, and $\mathcal{A}_i^{-1}$ is a sequence of actions of the form $(a_1'\otimes\ldots\otimes a_k')$. Then, the inversion of S is the transaction formula $\mathtt{Inv}(S)=\mathcal{A}_n^{-1}\otimes\ldots\otimes\mathcal{A}_1^{-1}$.*

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 11 (Rollback and Recovery)**.** *Recall the previous example 10 and consider the following path $\pi = \langle(\{\}, E_1),^{p.ins} (\{p\}, E_1),^{\mathbf{ext}(a,a_1\otimes a_2)} (\{p\}, E_2)\rangle$. Also, assume that $\mathcal{O}^e(E_2, E_3) \models a_1$ and $\mathcal{O}^e(E_3, E_4) \models a_2$. Then, the* rollback path *π_0 of π is the path as follows $\langle(\{\}, E_1),^{\mathbf{ext}(a,a_1\otimes a_2)} (\{\}, E_2)\rangle$. Additionally,* $\texttt{Seq}(\pi_0) = \langle \mathbf{ext}(a, a_1 \otimes a_2)\rangle$ *and* $\texttt{Inv}(\texttt{Seq}(\pi_0)) = a_1 \otimes a_2$. *Finally, $\langle(\{\}, E_2),^{a_1} (\{\}, E_3),^{a_2} (\{\}, E_4)\rangle$ 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 issued due to the execution of formula ϕ are compensated, and the internal state is rollbacked.

Definition 17 (Compensating Path for a Transaction)**.** *Let M be an interpretation, π a path and ϕ a formula. $M, \pi \rightsquigarrow \phi$ iff* 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. $\texttt{Seq}(\pi_1) \neq \emptyset$ *and $\exists\pi_r$ such that π_r is a recovery path of* $\texttt{Seq}(\pi_1)$ *w.r.t. M*
4. *π_0 and π_r are a split of π, i.e. $\pi = \pi_0 \circ \pi_r$*

Example 12 (Compensating Path)**.** *In the scenario of the previous examples 10 and 11, the following statement $M, \langle(\{\}, E_1),^{\mathbf{ext}(a,a_1\otimes a_2)} (\{\}, E_2),^{a_1} (\{\}, E_3),^{a_2} (\{\}, E_4)\rangle \rightsquigarrow p.ins \otimes \mathbf{ext}(a, a_1 \otimes a_2) \otimes \mathbf{ext}(b, b_1)$ 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.*

The previous definition retrieves paths where a formula ϕ is *not* successfully executed, but where external recovery can still be guaranteed. Consequently, notice that consistency preserving paths are only defined for cases where besides a primitive action fails, some external actions with compensations were executed. This is achieved since the operator $\texttt{Seq}(\pi_1)$ 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_1 or if all the external actions executed are not meant to be compensated (e.g. if they are external queries), then $\texttt{Seq}(\pi_1) = \emptyset$. 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 what (complex) formulas are true on what 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, $M, \pi \models \phi$.*

Note that the previous 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 (point 6). As expected, recovery only makes sense in situations where some external actions were performed before the failure. Otherwise we can just rollback for the initial state and try to satisfy the formula in an alternative branching.

Example 13. *Recall examples 10 and 11, and 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 $\langle(\{\}, E_1),^{q.ins} (\{q\}, E_1),^{\mathbf{ext}(c,c_1)} (\{q\}, E_5)\rangle$ (without using compensations), and also where the first rule is tried in $\langle(\{\}, E_1),^{\mathbf{ext}(a,a_1\otimes a_2)} (\{\}, E_2),^{a_1} (\{\}, E_3),^{a_2} (\{\}, E_4),^{q.ins} (\{q\}, E_4),^{\mathbf{ext}(c,c_1)} (\{q\}, E_5)\rangle$ (by using point 6).*

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 formalize this correspondence as follows.

Theorem 2. *Let M be an interpretation, ϕ an $\mathcal{ETR}$ formula and π, π' paths such that π' is a path where no external actions appear in the transitions. Then:*

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

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

After this, we can now define what is a model and the notion of logical entailment. Due to the restriction we imposed on $\mathcal{ETR}$'s use of negation, we alternatively define that an interpretation is a model of a rule if it models the disjunction of the bodies of that rule.

Definition 19 (Models and Logical Entailment)**.** *Let ϕ and ψ be two $\mathcal{ETR}$ formulas and M be an interpretation. M is a model of ϕ (denoted $M \models \phi$) iff $M, \pi \models \phi$ for every path π. Let $Bodies(A)$ be the disjunction of the bodies of all rules with head A. An interpretation M is a model of a program P if, for every atom $A \in \mathcal{L}_P$ and every path π, whenever $M, \pi \models Bodies(A)$ then $M, \pi \models A$.*
We say that ϕ logically entails ψ ($\phi \models \psi$) if every model of ϕ is also a model of ψ.

Based on the definition of a model, logical entailment specifies a general entailment that 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" ($\phi \models \psi \wedge \psi \models \phi$).

3.4. Executional Entailment

The previously defined general notion of entailment is very powerful since it considers *all* the paths of execution that satisfy a given formula. However, 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 to *execute* $\mathcal{ETR}$ programs. This entailment is inherit from $\mathcal{TR}$.

Definition 20 (Executional Entailment)**.** *Let P be a program, ϕ be a formula and $S_1,^{A_1} \ldots,^{A_{n-1}} S_n$ be a path:*

$$P, (S_1,^{A_1} \ldots,^{A_{n-1}} S_n) \models \phi \tag{4}$$

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

The expression $P, (S_1,^{A_1} \ldots,^{A_{n-1}} S_n) \models \phi$ denotes that given a transaction program P, the path $(S_1,^{A_1} \ldots,^{A_{n-1}} S_n)$ represents a valid execution for transaction ϕ. In our running example, both statements hold: $P, \langle(\{\}, E_1),^{\mathbf{ext}(a, a_1 \otimes a_2)} (\{\}, E_2),^{a_1} (\{\}, E_3),^{a_2} (\{\}, E_4),^{q.ins} (\{q\}, E_4),^{\mathbf{ext}(c, c_1)} (\{q\}, E_5)\rangle \models t$ and $P, \langle(\{\}, E_1),^{q.ins} (\{q\}, E_1),^{\mathbf{ext}(c, c_1)} (\{q\}, E_5)\rangle \models t$.

Further, the statement $P, S_1 - \models \phi$ 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).

Based on this notion of executional entailment, we can now precise the relation between $\mathcal{TR}$ and $\mathcal{ETR}$. Namely we can conclude that 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 valid both in $\mathcal{TR}$'s and in $\mathcal{ETR}$'s syntax. Then:*

$$P, \pi' \models_{TR} \phi \text{ iff } P, \pi \models_{ETR} \phi$$

where π' is obtained from π by removing the external component and the annotated transitions in every transition of states.

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

The previously defined 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 executing transactions in that way. In this section we extend the proof theory for the ground[3] 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

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

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.

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 have prove that ϕ can succeed starting from S_0, but we also found a path where ϕ succeeds.

Most of the derivation rules are equivalent to the ones of $\mathcal{TR}$'s proof theory and are pretty easy to follow: when proving an atom which is at the head of some rule, replace the atom in the resolvent by the body of the rule; when proving an atom that is true in an oracle, just remove the atom from the resolvent, and update the path and state appropriately. This forms the basis of the so called $SLD_{\mathcal{ETR}}$ classical derivation.

Moreover, contrarily to $\mathcal{TR}$ proof theory, $SLD_{\mathcal{ETR}}$ includes an additional rule to deal with compensations. For that, we need, besides the usual notion of successful derivation, to handle also with derivations that do not succeed, but end in the execution of an action that fails in the oracle. If that is the case, then we need to rollback, compensate the executed actions, and proceed. To handle derivations that fail we further specify the notion of *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 action primitive that cannot be executed in S_f.

Definition 21 ($SLD_{\mathcal{ETR}}$ 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 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 22 (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.* $\mathcal{O}^e(E_f, E_i) \models L_1$

Taken together, definitions 21 and 22 determine a sound and complete procedure 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 such 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.

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 $P, \pi \vdash \phi$.*

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

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 [31,14,27], 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. Contrarily, 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 knowl-

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 $\mathcal{O}^d(D_1) \models L_1$

$\mathbf{r_3}$. $\pi \circ \langle (D_1, E_1),^{L_1} (D_2, E_1) \rangle, (D_2, E_1) \Vdash_P L_2 \otimes \ldots \otimes L_n$ if $\mathcal{O}^t(D_1, D_2) \models L_1$

$\mathbf{r_4}$. $\pi \circ \langle (D_1, E_1),^{L_1} (D_1, E_2) \rangle, (D_1, E_2) \Vdash_P L_2 \otimes \ldots \otimes L_n$ if $\mathcal{O}^e(E_1, E_2) \models L_1$

$\mathbf{r_5}$. $\pi \circ \langle S_1,^{A_1} \ldots,^{A_{p-1}} S_p,^{A_k^{-1}} \ldots,^{A_1^{-1}} S_q \rangle, S_q \Vdash_P L_1 \otimes \ldots \otimes L_n$ if all conditions hold:

- 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 $\langle S_1,^{A_1} \ldots,^{A_{j-1}} S_j \rangle, S_j \Vdash_P \phi$, for some transaction ϕ
- $S_1,^{A_1} \ldots,^{A_{p-1}} S_p$ is the rollback path of $S_1,^{A_1} \ldots,^{A_{j-1}} S_j$ (cf. Definition 15)
- $\texttt{Inv}(\texttt{Seq}(\langle S_1,^{A_1} \ldots,^{A_{p-1}} S_p \rangle)) = A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ (cf. Definition 16)
- 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 $\langle S_p,^{A_k^{-1}} \ldots,^{A_1^{-1}} S_q \rangle, S_q \Vdash_P ()$

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

edge 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 [43,45,32] 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 [24,39,37] but also over these hybrid knowledge bases [52,53].

Based on the development of such operators, the following step is to provide transactional properties over such evolution of knowledge. In this sense, by providing means to reason and execute transactions defined over internal and external independent domains, $\mathcal{ETR}$ can be used to achieve this goal. In fact, the flexibility obtained by the inclusion of oracles allows $\mathcal{ETR}$ to be suitable, independently of which W3C standard is selected for the particular moment. Also, it should be noted that achieving different transactional properties over actions, depending on whether they are internal or external, is in line with the previous arguments for the combination of closed and open world reasoning.

However, for such Semantic Web usage, specific instantiations of the oracles are in order. With this goal, and since depending on the application Description Logics can be used to describe both the internal and external KB, next we start by providing an example of a Description Logic instantiation for these domains.

Additional semantics may also be useful in this context, as 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. With this in mind, in this section we also provide an instantiation for Action Languages, Situation Calculus, and Event Calculus as external oracle definitions.

4.1. Description Logics Oracles

With the goal to be useful in a Semantic Web context, we exemplify how the internal oracles can be defined for a Description Logic KB. Description Logics [1] have been largely used to describe knowledge in the Semantic Web and are the underlying representation formalism of the standard Web Ontology Language (OWL) [41].

As decidable subsets of first-order logic, Description Logics comprise several family languages with different expressivity and complexity features. Every Description Logic (DL) knowledge base $\mathcal{K}$ is composed by knowledge described over a TBox ($\mathcal{T}$) and a ABox ($\mathcal{A}$). Here, the TBox defines the concepts and terminologies of the world while the ABox defines assertions of particular instances.

Based on just this, we can abstractly define the database oracle $\mathcal{O}^d$ as a mapping from a DL knowledge base $\mathcal{K}$ to the set of formulas that are true in that state. Then, a state identifier D can be defined as the pair $\langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is a ABox, and $\mathcal{O}^d$ is such that a formula is true in it iff it is a consequence of the TBox plus the ABox.

Different oracle instantiations can be defined for different Description Logics. Here, for the purpose of this illustration, we provide an instantiation for the *DL-Lite* Family [11]. *DL-Lite* is the backbone of the OWL-2 QL profile [17] and known for its low computational complexity on large volumes of instance data (ABox size). OWL 2 [44] is the second edition of the standard OWL and is fragmented upon three different profiles with the goal to address different application requirements. In this sense, the OWL 2 QL 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 and thus to be more concrete let us pick *DL-Lite*$_{\mathcal{FR}}$ that enjoys from polynomial algorithms to update the ABox [24, 13].

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:

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

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, the ABox $\mathcal{A}$ is composed by a set of membership assertions of the form $B(a)$ and $P(a, b)$ where a and b are object constants. The TBox $\mathcal{T}$ is a set assertions of the following:

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

As any Description Logic, its semantics is defined by first-order logic interpretations. An interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ provides a non-empty set of interpretation domain Δ, and a mapping $\cdot^{\mathcal{I}}$ which maps individuals, concepts and roles to Δ as follows:

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

To simplify, we assume to have *standard names*, i.e. that there is no distinction between the alphabet of constants and Δ. Finally, an interpretation $\mathcal{I}$ is said to model a given TBox or ABox assertion F (denoted $\mathcal{I} \models F$) 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 satisfies a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ (written $\mathcal{I} \models \mathcal{K}$) iff it satisfies each assertion in $\mathcal{K}$. Finally, let $Mod(\mathcal{K})$ be the set of all models of $\mathcal{K}$. Then, $\mathcal{K}$ is *satisfiable* iff $Mod(\mathcal{K}) \neq \emptyset$ (i.e. if it has at least one model) and *unsatisfiable* otherwise.

4.1.1. DL-Lite database oracle

After specifying the background for *DL-Lite* we can now elaborate on a example for a database oracle defined for such Description Logic. For that we can make use of *DL-Lite* characteristics and employ a query-answering algorithm for conjunctive queries as defined in [12]. Here, a conjunctive query q over $\mathcal{K}$ is an expression of the form:

$$q(\vec{x}) \leftarrow \exists \vec{y}.conj(\vec{x}, \vec{y}) \tag{5}$$

where $\vec{x}$ and $\vec{y}$ are respectively known as the *distinguished variables* and *non-distinguished variables* of query q. The conjunctive query is defined by the formula $conj(\vec{x}, \vec{y})$ which denotes 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 $\vec{y}$.

Formula (5) above is interpreted in an interpretation $\mathcal{I}$ as the set of $q^{\mathcal{I}}$ 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 $\mathcal{I}$. Then, $\texttt{ans}(q(\vec{x}), \mathcal{K})$ is the set of such tuples such that $\exists \vec{y}.conj(\vec{x}, \vec{y})$ is true in every $\mathcal{I}$ that is a model of $\mathcal{K}$.

We say that a query $q(\vec{x})$ is true in the knowledge base $\mathcal{K}$ if $\texttt{ans}(q(\vec{x}), \mathcal{K}) \neq \emptyset$. If $\vec{x}$ is ground, then q is said to be *boolean*, and in such case $\texttt{ans}(q, \mathcal{K})$ consists of the empty tuple whenever q holds in every model of $\mathcal{K}$. Additionally, if $\mathcal{K}$ is unsatisfiable, then $\texttt{ans}(q(\vec{x}), \mathcal{K}))$ is the set of all possible tuples of constants in $\mathcal{K}$ with the same arity as $\vec{x}$.

To query with such a *DL-Lite* KB $\mathcal{K}$, we can define as primitive $\texttt{dlquery}(\vec{c}, conj(\vec{c}, \vec{y}))$ where $conj(\vec{c}, \vec{y})$ is a conjunctive query, and $\vec{c}$ the set of constants that appear in it. Then the database oracle $\mathcal{O}^d$ can be as follows.

Definition 23 (*DL-Lite* database oracle)**.** *Let $\mathcal{K}$ be a state (i.e. a TBox and an ABox in DL-Lite).*

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

Note that, with this definition, we are assuming all DL queries to be boolean. 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 the DL queries possibly appearing in them. For the general case of rules with variables, a condition similar to DL-safety [46] 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.

4.1.2. DL-Lite transition oracle

After specifying an oracle to query $\mathcal{ETR}$'s internal KB, we further exemplify on how this internal KB can be updated.

For that we restrict to instance-based updates, i.e. updates on the membership assertions of the ABox. Then, an update $\mathcal{U}$ is simply a set of ABox assertions that is integrated into the current knowledge base $\mathcal{K}$, achieving 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 only for 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 [39] 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 define $\texttt{dladd}()$ as the primitive to add a given update to a Description Logic. Then, the transition oracle can be defined as follows.

Definition 24 (*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 referenced earlier.*

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

4.2. External oracles

As stated, $\mathcal{ETR}$'s logic is parametric on the oracles $\mathcal{O}^d$, $\mathcal{O}^t$, and $\mathcal{O}^e$, that encapsulate a specific representation of states and describe, respectively, what holds in each state of the internal KB, the possible transitions of the internal KB and the external domain.

While instantiating the internal oracles is required to use $\mathcal{ETR}$ (or $\mathcal{TR}$), the external oracle can be left open if all we want is execute an $\mathcal{ETR}$ program. This is as expected since in most cases it is impossible to know how the external oracle is specified. However, to reason about general properties of $\mathcal{ETR}$ transactions, like equivalence or implication, then an external representation of states must be chosen and the external oracle must be properly defined. This leaves open the question of how this external oracle $\mathcal{O}^e$ can be instantiated and what semantics are useful to characterize the external KB.

In a Semantic Web environment the external KB 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 knowledge bases). Consequently, in the following we define an $\mathcal{O}^e$ for a DL-Lite Description Logic 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, the 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 is established by this variety of actions, one has to define external oracles for such languages. Because of that, we also provide a $\mathcal{O}^e$ specification for Action Languages, Situation Calculus, and Event Calculus.

4.2.1. External Description Logic Oracle

Defining a Description Logic external oracle is just equivalent to defining the internal oracles $\mathcal{O}^d$ and $\mathcal{O}^t$ for a given DL. Yet, since $\mathcal{O}^e$ works always with pairs of states even when evaluating queries, then 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. $\mathcal{O}^e(\mathcal{K}, \mathcal{K})$.

Moreover, it may make sense to modify such oracle for a scenario where the intentions of the involved parties is unknown. 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 with 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. Based on this, an external oracle for a *DL-Lite* DL can be defined as follows:

1. $\mathcal{O}^e(\mathcal{K}, \mathcal{K}) \models \mathtt{dlquery}(c, \phi(c, \vec{x}))$ iff $q \leftarrow \phi(c, \vec{x}) \wedge ans(q, \mathcal{K}) \neq \emptyset$
2. $\mathcal{O}^e(\langle \mathcal{T}, \mathcal{A} \rangle, \langle \mathcal{T}, \mathcal{A}' \rangle) \models \mathtt{dladd}(\mathcal{U})$ iff $\mathcal{A}' = \mathtt{c\_upd}(\mathcal{T}, \mathcal{A}, \mathcal{U}) \wedge safe(\mathcal{A}, \mathcal{A}', List(User))$

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.

As previously argued, other languages and domains are also possible in the external KB. Thus, in the following we illustrate how the external oracle can be defined for languages with the aim of encoding the dynamic effects of actions, as Action Languages, and Situation and Event Calculus.

4.2.2. Action Language Oracle

Action Languages are a family of languages proposed in [23] with the goal to model the dynamics of external environments.

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 $\langle \{\mathtt{true}, \mathtt{false}\}, \mathcal{F}, \mathcal{A} \rangle$ 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 $\{\mathtt{true}, \mathtt{false}\}$, 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. Thend 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 $fluent \in \mathcal{F} \wedge V(fluent, s) = \mathtt{true}$

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: $\text{Effects}_E(A, s) = \{L : there\ exists\ A$ **causes** L **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 [23].

4.2.3. Situation Calculus Oracle

In the seminal Situation Calculus [40], 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 [48] 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 14 (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:

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

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.

4.2.4. Event Calculus Oracle

Similarly, in Event Calculus [34] predicates can be *actions* or *fluents*, where actions can change properties of the world and fluents denote these properties whose truth value may change. The main innovation 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[4]: $initially(f)$, 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:

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

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 instantiate 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 [28], one can express the possibility of killing turkey Fred as follows:

$$\begin{array}{r} executable(kill, T) \leftarrow holdsAt(alive, T), \\ holdsAt(loaded, T). \end{array}$$

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 15 (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

[4]We assume the simplified version of the calculus as defined in [51]. More basic predicates can be found in the full version of the calculus.

slopes and if the resort is open or not.

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

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

$holdsAt(isClosed, T) \leftarrow holdsAt(stormStart, T_1),$
$\quad T_1 \leq T, T_1 \leq T_2 \leq T, \mathbf{not}\ holdsAt(stormEnd, T_2).$
$holdsAt(isOpen, T) \leftarrow \mathbf{not}\ holds(isClosed, T).$
$holdsAt(stormStart, 150313).$
$holdsAt(stormEnd, 180313).$
$holdsAt(snowCM(10, 150313).$
$holdsAt(snowCM(150, 183113).$

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 consider external oracles described by a single semantics. However, the previously example 15 exemplifies a situation where more than one external semantics is required to describe the external domain. Particularly, in such 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 15, 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 and external KB defined by an abstract semantics.

In this sense, logics like Action Languages [23], the Situation Calculus [40], the Event Calculus [34], Process Logic [29], etc.; and some of their variations like [19,38,33,35] 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 receives 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. 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 retrieve the correct compensation of a given action according to a domain description. This notion was further developed in [26] for an action language oracle. Here 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 [10,55,3]. 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 [42] or Hoare's CSP [30], 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, these solutions 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 [55]. 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 with 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 [16]. 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, contrarily to $\mathcal{ETR}$ this notion is not reflected in ULTRA's model theory which does not provide means to soften the ACID transactional model. Thus there is no formal correspondence between the procedure of ULTRA and its model theory as in $\mathcal{ETR}$.

In [49] 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. Nevertheless, $\mathcal{TR}^{PAD}$ goes against some of the original ideas of $\mathcal{TR}$. Particularly, the information about the domain and actions is defined directly in the logic, the notion of oracles is abandoned and states can only be defined by ground propositional formulas. This swift of paradigm makes the theory less flexible as it precludes the possibility of changing the semantics of states or combine several semantics to reason with more than one domain. It is also unclear how the transaction properties of $\mathcal{TR}$ can benefit the $\mathcal{TR}^{PAD}$ or how can they be ensured in an external domain like defined in $\mathcal{ETR}$. In addition, it is impossible to relax some properties of transactions as in $\mathcal{ETR}$.

Statelog [36] is a logic-programming like language with support for ACID 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 hardwired 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 ACID properties.

Finally, as a Semantic Web related solution we reference the RDF Triggering Language (RDFTL) [47]. 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 for $\mathcal{ETR}$, an extension of Transaction Logic to reason and execute (trans)actions performed both in an internal and an external knowledge base. To accommodate this, $\mathcal{ETR}$ guarantees different kinds of properties according to the domain where the actions were executed. Then, when executed in an internal KB, actions are guaranteed to follow the ACID model of transactions, meaning that either the transaction can succeed completely, or the execution is considered to have never happened. In opposition, when executed in an external KB, actions follow a relaxed version of this ACID model, achieved using compensations. Since external actions are executed in a domain in which, in principle, we cannot rollback actions, then it is no longer possible to guarantee that the external KB will remain the same after some failure. As an alternative to restore external consistency, $\mathcal{ETR}$ issues a sequence of compensating operations whenever a failure occurs after the execution of external actions.

With a model-theoretic semantics and a sound and complete top-down procedure, $\mathcal{ETR}$ is useful for executing *and* reasoning about systems that update both an internal and external component and need to guarantee properties on the outcome of these updates.

Examples of such systems are commonly found on the web, as e.g. a web-service with an internal database interacting with another web-service, or a web-source with a knowledge described by a Description Logic that needs to consult and execute actions in other web-sources.

An important feature of $\mathcal{ETR}$ is that its semantics of states and primitive operations is *abstract*. I.e. $\mathcal{ETR}$ allows execution and reasoning of transactions independently of the semantics chosen for the internal and external KB. This is achieved by defining oracles as a parameter of the theory. These oracles define the most appropriate semantics for the application in question, allowing $\mathcal{ETR}$ to be useful in a wide range of domains.

Since $\mathcal{ETR}$ "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. Then, 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 materialize the changes of a given transaction according to the model-theory, and find a particular path where the transaction is successfully executed (if that 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] 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.

[17] 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.

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

[19] 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.

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

[21] 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.

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

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

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

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

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

[31] 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.

[32] 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.

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

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

[35] Kim Guldstrand Larsen and Bent Thomsen. A modal process logic. In *LICS*, pages 203–210. IEEE Computer Society, 1988.

[36] 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.

[37] 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.

[38] Hector J. Levesque, Raymond Reiter, Yves Lespérance, Fangzhen Lin, and Richard B. Scherl. Golog: A logic programming language for dynamic domains. *J. Log. Program.*, 31(1-3):59–83, 1997.

[39] 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.

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

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

[42] Robin Milner. Calculi for synchrony and asynchrony. *Theor.*

Comput. Sci., 25:267–310, 1983.

[43] 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.

[44] 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.

[45] Boris Motik and Riccardo Rosati. A faithful integration of description logics with logic programming. In *IJCAI*, pages 477–482, 2007.

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

[47] 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.

[48] Raymond Reiter. The frame problem in situation the 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.

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

[50] Dumitru Roman and Michael Kifer. Reasoning about the behavior of semantic web services with concurrent transaction logic. In *VLDB*, pages 627–638, 2007.

[51] Murray Shanahan. The event calculus explained. In *AI Today*, pages 409–430. 1999.

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

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

[54] Michael Thielscher. Introduction to the fluent calculus. *Electron. Trans. Artif. Intell.*, 2:179–192, 1998.

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

Appendices

Auxiliary Results

Lemma 1. *Let M be an interpretation, π a path and ϕ an atom*

$$M, \pi \models \phi \textit{ iff } \phi \in M(\pi)$$

Proof. If ϕ is defined in the oracles languages, then this result is trivial by Definition 10. So let's assume that ϕ is not defined in the oracles. Then $M, \pi \models \phi$ if one of the two cases occur:

1. $\phi \in M(\pi)$ and thus the proof is complete, or;
2. $\exists \pi_1 \circ \pi_2$ splits of π s.t. $M, \pi_1 \rightsquigarrow \phi$ and $M, \pi_2 \models \phi$.
   We will prove that $M, \pi_1 \rightsquigarrow \phi$ is always false for these cases, and for that let's consider the definition of $\rightsquigarrow$. If $M, \pi_1 \rightsquigarrow \phi$ then it exists a split $\pi_0 \circ \pi_r$ of π_1 s.t. $M, \pi_1' \models_p \phi$ and $M, \pi_1' \not\models_c \phi$, and π_0 is the rollback path of π_1'. However, since ϕ is an atomic formula but does not belong to the language of the oracles, then $M, \pi_1' \models_p \phi$ and $M, \pi_1' \not\models_c \phi$ is impossible by the definition of the base case of the partial satisfaction. Then since $M, \pi_1 \rightsquigarrow \phi$ does not hold, then $M, \pi \models \phi$ can only hold by the the first case and thus $M, \pi \models \phi$ iff $\phi \in M(\pi)$.

□

Lemma 2. *Let M be an interpretation, and π be a path such that for every annotated action A_i that appears in π, A_i is **not** an external action. Then for every formula ϕ, $M, \pi \not\rightsquigarrow \phi$*

Proof. For $M, \pi \rightsquigarrow \phi$ to be true, then it must exist a recovery path π_r and a $S_M(\pi_r) \neq \emptyset$. If π does not contain external actions, then it is impossible to have the previous two conditions and thus $M, \pi \rightsquigarrow \phi$ for any formula ϕ □

Lemma 3. *Let P be a program and π a path. If $head \leftarrow body \in P$ and $M \models P$ then $M, \pi \models_c body$ implies $M, \pi \models_c head$*

Proof. Note that if $M \models P$, $head \leftarrow body \in P$ and $M, \pi \models_c body$ implies that $M, \pi \models body$ and $M, \pi \models head$. By definition of P, $head \in \mathcal{L}_P$ and thus by Lemma 1 we know that $M, \pi \models head$ iff $head \in M(\pi)$. Based on this, we can conclude that $M, \pi \models_c head$ □

Lemma 4. *Let P be a transaction program and ϕ a transaction formula. If ϕ and P do not contain external actions then: $P, \pi \models \phi$ only if π does not contain external actions*

Proof. $P, \pi \models \phi$ iff for all models M of P, $M, \pi \models \phi$. Since this holds for every model, then it must also hold for the model M_s, the *skeptical model* that only makes true the actions defined in paths of the oracle, and the heads of the rules in P where the bodies are necessarily true w.r.t. paths.

Since there are no external actions in P, then for any $head$ of rules in P, $M_s, \pi \models head$ is only true in paths where π does not contain external actions. Consequently, for any formula ϕ such that ϕ is not an external action and P does not have external actions in the body, $M_s, \pi \models \phi$ only in paths where π does not have external actions. □

Lemma 5. *Let P be a a transaction program both valid in $\mathcal{TR}$ and $\mathcal{ETR}$. Let M be an interpretation s.t. M is a valid interpretation in $\mathcal{ETR}$. Then M is a model in $\mathcal{TR}$ of P then M is a model in $\mathcal{ETR}$ of P.*

Proof. M is a model in $\mathcal{TR}$ of P iff M models every rule $h \leftarrow body$ of P. I.e. for every path π' $M, \pi' \models_{TR} \neg body$ or $M, \pi' \models_{TR} head$. This is equivalent to say that whenever $M, \pi' \models_{TR} body$ holds then $M, \pi' \models_{TR} head$ must hold as well. Note that this corresponds with the definition of model of $\mathcal{ETR}$. Furthermore we know that since P does not contain external actions, then for every path π_1 $M, \pi_1 \models_{ETR} body$ only if π_1 does not contain external actions (by Lemma 4). Let π be the path obtained from π' adding the constant state E_0. Then it is easy to see that $M, \pi' \models_{TR} \phi$ iff $M, \pi' \models_c \phi$ as both relations coincide with the definitions. Since π' does not contain external then by Equation 3 of Theorem 2 we know that $M, \pi' \models_{TR} \phi$ iff $M, \pi \models_{ETR} \phi$. □

Lemma 6. *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 $M, \pi_2 \not\models_c \phi$ then $M, \pi \not\models_c \psi \otimes \phi$*

Proof. Assume $M, \pi \models_c \psi \otimes \phi$ then $\exists$ a split π', π'' s.t. $M, \pi' \models_c \psi$ and $M, \pi'' \models_c \phi$. Since ψ is an atomic action then such split must be $\pi_1 \circ \pi_2$ and thus since $M, \pi_2 \not\models_c \phi$ then the assumption fails. □

Definition 25 (Unfolding of formulas). *Let P be a serial-Horn program and ϕ is a serial-goal of the form: $\phi = \phi_1 \otimes \ldots \otimes \phi_i \otimes \ldots \otimes \phi_k$. The unfolding of ϕ is repeatedly processed until it is complete as follows: For every atomic formula ϕ_i, if $\phi_i \leftarrow body \in P$ then $\phi = \phi_1 \otimes \ldots \otimes body \otimes \ldots \otimes \phi_k$. The unfolding is said to be complete if every ϕ_i is an action formula defined in the oracles.*

Theorem 5 (Soundness of $\mid\!\leadsto$). *Let P be a serial-Horn program, π a path, ϕ a serial-Horn goal and M an interpretation such that $M \models P$. Let $\phi_1 \otimes \ldots \otimes \phi_i$ be the complete unfold of formula ϕ w.r.t. P. If $P, \pi \mid\!\leadsto \phi_1 \otimes \ldots \otimes \phi_i$ then $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_i \wedge M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_i$*

Proof. From Lemma 8 and 9 □

Lemma 7. *Let ϕ be a completely unfolded serial-Horn goal, and M an interpretation s.t. $M \models P$. If $M, \pi \models_p \phi \wedge M, \pi \not\models_c \phi$, then $\forall \psi$ s.t. ψ is a serial-Horn goal, then $M, \pi \models_p \phi \otimes \psi \wedge M, \pi \not\models_c \phi \otimes \psi$*

Proof. **Base Case, k = 1:**
Let ϕ be an atom. Then for the statement to be true, π must be a 1-path. By definition of $\models_p$ it comes 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 $\pi \circ \pi$ and thus it follows that $M, \pi \not\models_c \phi \otimes \psi$.
Inductive Case:
Assume it is true for $\phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$. Then we want to prove it is still true for $\phi_1 \otimes \ldots \otimes \phi_{k+1}$. So assume that $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_{k+1} \wedge M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_{k+1}$. 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 can partition this in two cases:
(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 and then by I.H. we know that $\forall \psi_1$ s.t. ψ 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 is valid for any formula ψ_1 s.t. ψ 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 $\forall \pi_3 \circ \pi_4 = \pi. \; M, \pi_3 \not\models_c \phi_1 \otimes \ldots \otimes \phi_k \vee M, \pi_4 \not\models_c \phi_{k+1}$. Then, since $M, \pi_1 \models_c \phi_1 \otimes \ldots \otimes \phi_k$ then it must be true that $M, \pi_2 \models_p \phi_{k+1}$ and $M, \pi_2 \not\models_c \phi_{k+1}$. Consequently, we are in the base case that was already proven, and thus we can conclude that $M, \pi_2 \not\models_c \phi_{k+1} \otimes \psi$. Finally, since $M, \pi_1 \models_c \phi_1 \otimes \ldots \otimes \phi_k$ and every ϕ_i is an atom, then $M, \pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_k \otimes \phi_{k+1} \otimes \psi$ (as the split where $\phi_1 \otimes \ldots \otimes \phi_k$ is true is fixed to π_1, and in any other split ϕ_{k+1} does not hold). Note that $M, \pi \models_p \phi_1 \otimes \ldots \otimes \phi_k \otimes \psi$ comes immediately by definition of the serial conjunction case of $\models_p$. □

Lemma 8 (Soundness of $\mid\!\leadsto$ Axiom). *If $P, \pi \mid\!\leadsto L_1 \otimes \ldots \otimes L_k$ then $M, \pi \models_p L_1 \otimes \ldots \otimes L_k$ and $M, \pi \not\models_c L_1 \otimes \ldots \otimes L_k$*

Proof. **Base Case** $k = 1$
If $k = 1$, this case is only true for 1-paths $\langle S_f \rangle = \langle (D_f, E_f) \rangle$. Thus, since $P, \langle (D_f, E_f) \rangle \mid\!\leadsto L_1 \otimes \ldots \otimes L_k$ then the derivation starts and ends in the resolvent $\langle (D_f, E_f) \rangle, (D_f, E_f) \Vdash_P L_1 \otimes \ldots \otimes L_k$ and one of the two cases occur:

(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. $\mathcal{O}^e(E_f, E_i) \models L_1$

Since L_1 belongs to the language of the oracle, then it means that, for any M, then $L_1 \notin M, \langle (D_f, E_f) \rangle$ and that $\neg \exists D_i, E_f$ s.t. $L_1 \in M \langle (D_f, E_f),^{L_1} (D_i, E_f) \rangle$ or $L_1 \in M \langle (D_f, E_f),^{L_1} (D_f, E_i) \rangle$. As a result, by definition of the classical and partial satisfactions, $M, \langle (D_f, E_f) \rangle \not\models_c L_1$ although $M, \langle (D_f, E_f) \rangle \models_p L_1$.
Inductive Case:
Assume it is true for $L_1 \otimes \ldots \otimes L_k$, then it must also be true for $L_1 \otimes \ldots \otimes L_{k+1}$. If $P, \pi \mid\!\leadsto L_1 \otimes \ldots \otimes L_{k+1}$ then it exists a resolvent $P, \langle S_1, \ldots, S_f \rangle, S_f \Vdash_P L_i \otimes L_{k+1}$ where $(1 \leq i \leq k+1)$. Then there are two cases:
(1) $(1 \leq i \leq k)$ and thus $P, \pi \mid\!\leadsto L_1 \otimes \ldots \otimes L_k$ which by I.H. we know that $M, \pi \models_p L_1 \otimes \ldots \otimes L_k$ and $M, \pi \not\models_c L_1 \otimes \ldots \otimes L_k$ and by Lemma 7 we know that $M, \pi \models_p L_1 \otimes \ldots \otimes L_k \otimes L_{k+1}$ and $M, \pi \not\models_c L_1 \otimes \ldots \otimes L_k \otimes L_{k+1}$.
(1) $(i = k + 1)$ and thus it exists a classical derivation starting in $\langle S_1 \rangle, S_1 \Vdash_P L_1 \otimes \ldots \otimes L_k$ and ending in $\pi, S_f \Vdash_P ()$. Then by Lemma 19 $M, \pi \models_c L_1 \otimes \ldots \otimes L_k$. Moreover, we also have a action-failed derivation starting in $\langle S_f \rangle, S_f \Vdash_P L_k$ and ending in $\langle S_f \rangle, S_f \Vdash_P L_k$ and thus (as proven in the base case), $M, \langle S_f \rangle \models_p L_k$ and $M, \langle S_f \rangle \not\models_c L_k$. Then, since $\pi = \langle S_1,^{A_1} \ldots,^{A_{f-1}} S_f \rangle$ we can conclude that $M, \pi \models_p L_1 \otimes L_k \otimes L_{k+1}$ and $M, \pi \not\models_c L_1 \otimes L_k \otimes L_{k+1}$ (as the split for $\models_c$ is fixed and $M, \langle S_f \rangle \not\models L_{k+1}$). □

Lemma 9 (Soundness of $\mid\!\leadsto$ Rules). *We prove for each case individually:*

1. *We don't need to prove for this case since by definition the formula is completely unfolded and thus the rule is not applicable.*
2. *Assume that $\mathcal{O}^d(D_1) \models L_1$:*
   If $M, \pi \not\models_c L_2 \otimes \ldots \otimes L_k$ and $M, \pi \models_p L_2 \otimes \ldots \otimes L_k$ then
   $M, \pi \not\models_c L_1 \otimes \ldots \otimes L_k$ and $M, \pi \models_p L_1 \otimes \ldots \otimes L_k$
3. *Assume that $\mathcal{O}^d(D_1, D_2) \models L_1$:*
   If $M, \langle (D_2, E_2),^{A_2} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \not\models_c L_2 \otimes \ldots \otimes L_k$ and $M, \langle (D_2, E_2),^{A_2} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models_p L_2 \otimes \ldots \otimes L_k$ then $M, \langle (D_1, E_2),^{L_1} (D_2, E_2),^{A_2} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \not\models_c L_1 \otimes \ldots \otimes L_k$ and $M, \langle (D_1, E_2),^{L_1} (D_2, E_2),^{A_2} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models_p L_1 \otimes \ldots \otimes L_k$
4. *Assume that $\mathcal{O}^e(E_0, E_1) \models L_1$:*
   If $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \not\models_c L_2 \otimes \ldots \otimes L_k$ and $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models_p L_2 \otimes \ldots \otimes L_k$ then
   $M, \langle (D_1, E_0),^{L_1} (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \not\models_c L_1 \otimes \ldots \otimes L_k$ and $M, \langle (D_1, E_0),^{L_1} (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models_p L_1 \otimes \ldots \otimes L_k$

Proof. We prove each rule individually:

1. Not applicable
2. Assume $\mathcal{O}^d(D_1) \models L_1$
   If $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \not\models_c L_2 \otimes \ldots \otimes L_k$ and
   $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models_p L_2 \otimes \ldots \otimes L_k$ then
   $M, \langle (D_1, E_1),^{A_1}, \ldots,^{A_{f-1}} (D_f, E_f)\rangle \not\models_c L_1 \otimes \ldots \otimes L_k$ and
   $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models_p L_1 \otimes \ldots \otimes L_k$:
   We know that since $\mathcal{O}^d(D_1) \models L_1$ then for all interpretations M, $L_1 \in M(\langle (D_1, E_1)\rangle)$. Thus by the serial conjunction case of the partial satisfaction definition we can conclude that $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models_p L_1 \otimes \ldots \otimes L_k$. Moreover, by Lemma 18 we can conclude that $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \not\models_c L_1 \otimes \ldots \otimes L_k$
3. Assume $\mathcal{O}^t(D_0, D_1) \models L_1$
   If $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \not\models_c L_2 \otimes \ldots \otimes L_k$ and
   $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models_p L_2 \otimes \ldots \otimes L_k$ then
   $M, \langle (D_0, E_1),^{L_1} (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \not\models_c L_1 \otimes \ldots \otimes L_k$ and $M, \langle (D_0, E_1),^{L_1} (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models_p L_1 \otimes \ldots \otimes L_k$:
   Since $\mathcal{O}^t(D_0, D_1) \models L_1$ then for all interpretations $M, L_1 \in M(\langle (D_0, E_1),^{L_1} (D_1, E_1)\rangle)$. Thus by the serial conjunction case of the partial satisfaction definition we can conclude that $M, \langle (D_0, E_1),^{L_1} (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models_p L_1 \otimes \ldots \otimes L_k$. Finally, by Lemma 18 we conclude that: $M, \langle (D_0, E_1),^{L_1} (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \not\models_c L_1 \otimes \ldots \otimes L_k$
4. Assume $\mathcal{O}^e(E_0, E_1) \models L_1$:
   If $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \not\models_c L_2 \otimes \ldots \otimes L_k$ and
   $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models_p L_2 \otimes \ldots \otimes L_k$ then
   $M, \langle (D_1, E_0),^{L_1} (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \not\models_c L_1 \otimes \ldots \otimes L_k$ and $M, \langle (D_1, E_0),^{L_1} (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models_p L_1 \otimes \ldots \otimes L_k$:
   Since $\mathcal{O}^e(E_0, E_1) \models L_1$ then for all interpretations $M, L_1 \in M(\langle (D_1, E_0),^{L_1} (D_1, E_1)\rangle)$. Thus by the serial conjunction case of the partial satisfaction definition we can conclude that $M, \langle (D_1, E_0),^{L_1} (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models_p L_1 \otimes \ldots \otimes L_k$. Finally, by Lemma 18 we conclude that: $M, \langle (D_1, E_0),^{L_1} (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \not\models_c L_1 \otimes \ldots \otimes L_k$

□

Lemma 10 (Weakening of $\mid\!\leadsto$). *Let P be a serial-Horn program, ϕ be a serial-Horn formula, and $S_1,^{A_1} \ldots,^{A_{f-1}} S_f$ a sequence.*

If $P, S_1,^{A_1} \ldots,^{A_{f-1}} S_f \mid\!\leadsto \phi$ and it is a complete derivation, then for any serial-Horn formula ψ it holds $P, S_1,^{A_1} \ldots,^{A_{f-1}} S_f \mid\!\leadsto \phi \otimes \psi$

Proof. Immediately by definition of the proof procedure. □

Lemma 11. *Let M be an interpretation, π_1,π_2 a path such that $\pi_1 \circ \pi_2$ and ϕ, ψ a complete unfolded formula. If $M \models P$, $M, \pi_1 \models_c \phi$ and $P, \pi_2 \mid\!\leadsto \psi$ then $P, \pi_1 \circ \pi_2 \mid\!\leadsto \phi \otimes \psi$*

Proof. Immediate by definition of $\models_c$ and definition of the procedure. □

Theorem 6 (Completeness of $\mid\!\leadsto$). *Let P be a serial-Horn program, π a path, and M an interpretation such that $M \models P$. Let $\phi_1 \otimes \ldots \otimes \phi_i$ be the complete unfolded formula.*

$$\textit{If } M,\pi \models_p \phi_1 \otimes \ldots \otimes \phi_k \wedge M,\pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_k$$
$$\textit{then } P,\pi \mathrel{|\!\!\leadsto} \phi_1 \otimes \ldots \otimes \phi_k$$

Proof. We prove by induction on the length of the serial $\phi_1 \otimes \ldots \otimes \phi_k$. Note that k must be greater than 0 as it is impossible for the empty transaction to find a π s.t. $M,\pi \models_p ()$ but $M,\pi \not\models_c ()$

Base Case $k = 1$:
$M,\pi \models_p \phi_1$ and $M,\pi \not\models_c \phi_1$. Then since ϕ_1 is an atomic action defined in the oracles, then ϕ_1 is completely fixed in the interpretations. Thus, this statement is only true where π is a 1-path $\langle(D,E)\rangle$ such that $M,\langle(D,E)\rangle \not\models \phi_1$ and since $M,\langle(D,E)\rangle \models_p \phi_1$ then either ϕ_1 is an internal action and thus $\neg\exists D_i$ such that $\mathcal{O}^t(D,D_i) \models \phi_1$ or $\mathcal{O}^d(D) \not\models \phi_1$. If ϕ_1 is an external action then it must be the case that $\neg\exists E_i s.t. \mathcal{O}^e(E,E_i) \models \phi_1$.
Finally by definition 22 we have that $P,\langle(D,E)\rangle \mathrel{|\!\!\leadsto} \phi_1$

Inductive Case $k = j+1$:
Let's assume that this is true for $\phi_1 \otimes \ldots \otimes \phi_j$. We will prove that this remains true for $\phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$.
So, $M,\pi \models_p \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1} \wedge M,\pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$. Then one of the two cases occur:

1. $\exists \pi_1,\pi_2$ s.t. $\pi = \pi_1 \circ \pi_2$ and:
$M,\pi_1 \models_c \phi_1 \otimes \ldots \otimes \phi_j$, $M,\pi_2 \models_p \phi_{j+1}$ and $M,\pi_2 \not\models_c \phi_{j+1}$. Then since ϕ_{j+1} has size 1, then we can use the proof of the base case and infer $P,\pi_2 \mathrel{|\!\!\leadsto} \phi_{j+1}$. Moreover by Lemma 11 we have that $P,\pi_1 \circ \pi_2 \mathrel{|\!\!\leadsto} \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$
2. $M,\pi \not\models_c \phi_1 \otimes \ldots \otimes \phi_j \wedge M,\pi \models_p \phi_1 \otimes \ldots \otimes \phi_j$. Since this implies that $P,\pi \mathrel{|\!\!\leadsto} \phi_1 \otimes \ldots \otimes \phi_j$ then by Lemma 10 we have that $P,\pi \mathrel{|\!\!\leadsto} \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$

□

Lemma 12. *Let P be a serial-Horn program, M be a model of P, π be a path, ϕ a serial goal. Then:*

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

Proof. **Soundness of Axiom**:
If $\phi = ()$ then for any 1-path π, $P,\pi \vdash_c ()$. Moreover, since $()$ represents the empty transaction which is tautologically true in any path of length 1.
Soundness of Rules:
We prove each rule in turn:

1. Assume it exists a rule $L_1 \leftarrow B_1 \otimes \ldots \otimes B_j$ in P. Then we want to prove that if $M,\pi \models_c B_1 \otimes \ldots \otimes B_j \otimes L_2 \otimes \ldots \otimes L_k$ then $M,\pi \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$
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 it exists 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, then by Lemma 3 we have that $M,\pi_1 \models_c L_1$ and thus $M,\pi \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$
2. Assume that $\mathcal{O}^d(D_1) \models L_1$. Then we want to prove that if $M,\langle(D_1,E_1),^{A_1} \ldots,^{A_f} (D_f,E_f)\rangle \models_c L_2 \otimes \ldots \otimes L_k$ then it holds $M,\langle(D_1,E_1),^{A_1} \ldots,^{A_f} (D_f,E_f)\rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$.
Since $\mathcal{O}^d(D_1) \models L_1$, then 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 $\models_c$ we can conclude that
$M,\langle(D_1,E_1),^{A_1} \ldots,^{A_f} (D_f,E_f)\rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$
3. Assume that $\mathcal{O}^t(D_0,D_1) \models L_1$. Then we want to prove that if $M,\langle(D_1,E_1),^{A_1} \ldots,^{A_f} (D_f,E_f)\rangle \models_c L_2 \otimes \ldots \otimes L_k$ then it also holds that $M,\langle(D_0,E_1),^{L_1} (D_1,E_1),^{A_1} \ldots,^{A_f} (D_f,E_f)\rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$.
Since $\mathcal{O}^t(D_0,D_1) \models L_1$, we know that for every interpretation: $M,\langle(D_0,E_1),^{L_1} (D_1,E_1)\rangle \models_c L_1$. Thus, by definition of the serial conjunction case of $\models_c$ we can conclude that
$M,\langle(D_0,E_1),^{L_1} (D_1,E_1),^{A_1} \ldots,^{A_f} (D_f,E_f)\rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$
4. Assume that $\mathcal{O}^e(E_0,E_1) \models L_1$. Then we want to prove that if $M,\langle(D_1,E_1),^{A_1} \ldots,^{A_f} (D_f,E_f)\rangle \models_c L_2 \otimes \ldots \otimes L_k$ then it also holds $M,\langle(D_1,E_0),^{L_1} (D_1,E_1),^{A_1} \ldots,^{A_f} (D_f,E_f)\rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$.
Since $\mathcal{O}^e(E_0,E_1) \models L_1$, we know that for every interpretation: $M,\langle(D_1,E_0),^{L_1} (D_1,E_1)\rangle \models_c L_1$. Thus, by definition of the serial conjunction case of $\models_c$ we can conclude that
$M,\langle(D_1,E_0),^{L_1} (D_1,E_1),^{A_1} \ldots,^{A_f} (D_f,E_f)\rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$

□

Lemma 13. *Let P be a serial-Horn program, M be a model of P, π be a path, ϕ a completely unfolded serial-goal. Then the following is true:*

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

Proof. Proof by induction on the length of $\phi = \phi_1 \otimes \ldots \otimes \phi_k$.
Base Case k = 0
If $k = 0$ then $\phi = ()$. Thus $M,\pi \models_c ()$ for any 1-

path π. By definition, $P, \pi \models_c ()$ also holds and it is complete.

Inductive Case k = j+1

We assume that the statement is true for all values up to j and prove that it still holds for values $j+1$. We start by assuming that $M, \pi \models_c \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ is true. Then we know by the serial conjunction case of $\models_c$ that it exists a split $\pi_1 \circ \pi_2$ s.t. $M, \pi_1 \models_c \phi_1 \otimes \ldots \otimes \phi_j$ and $M, \pi_2 \models \phi_{j+1}$. Now since ϕ is a completely unfolded formula are three possible cases for ϕ_{j+1}:

1. $\pi_2 = \langle (D,E) \rangle \wedge \mathcal{O}^d(D) \models \phi_{j+1}$ then for any state E, by rule 2. of Definition 21 we have that $P, \langle (D,E) \rangle \vdash_c \phi_{j+1}$
2. $\pi_2 = \langle (D,E),^{\phi_{j+1}} (D_1,E) \rangle \wedge \mathcal{O}^t(D,D_1) \models \phi_{j+1}$ then for any state E, by rule 2. of Definition 21 we have that $P, \langle (D,E),^{\phi_{j+1}} (D_1,E) \rangle \vdash_c \phi_{j+1}$
3. $\pi_2 = \langle (D,E),^{\phi_{j+1}} (D,E_1) \rangle \wedge \mathcal{O}^e(E,E_1) \models \phi_{j+1}$ then for any state E, by rule 2. of Definition 21 we have that $P, \langle (D,E),^{\phi_{j+1}} (D,E_1) \rangle \vdash_c \phi_{j+1}$

Moreover, since $P, \pi_1 \vdash_c \phi_1 \otimes \ldots \otimes \phi_j$ and $P, \pi_2 \vdash_c \phi_{j+1}$ then since ϕ_2 starts in the last state of π_1 and $P, \pi_2 \vdash_c \phi_{j+1}$ is complete, then $P, \pi_1 \circ \pi_2 \vdash_c \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ □

Lemma 14 (Soundness & Completeness rule-5. w.r.t. $\rightsquigarrow$)**.** *Let P be a serial-Horn program, $L_1 \otimes \ldots \otimes L_k$ be a completely unfolded serial-Horn goal. Let the following three conditions be true:*

- $P, \langle S_1,^{A_1} \ldots,^{A_{j-1}} S_j \rangle \rightsquigarrow L_1 \otimes \ldots \otimes L_n$
- $S_1,^{A_1} \ldots,^{A_{p-1}} S_p$ *is the rollback path of the path* $S_1,^{A_1} \ldots,^{A_{j-1}} S_j$ *(cf. Definition 15)*
- $\texttt{Inv}(\texttt{Seq}(\langle S_1,^{A_1} \ldots,^{A_{p-1}} S_p \rangle)) = A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ *(cf. Definition 16)*
- $P, \langle S_p,^{A_k^{-1}} \ldots,^{A_1^{-1}} S_q \rangle, S_q \vdash_c A_k^{-1} \otimes \ldots \otimes A_1^{-1}$

Then, for every model M of P it holds that:
$M, \langle S_1,^{A_1} \ldots,^{A_{p-1}} S_p,^{A_{p-1}^{-1}} \ldots,^{A_1^{-1}} S_q \rangle \rightsquigarrow L_1 \otimes \ldots \otimes L_k$

Proof. Immediate since $P, \langle S_1,^{A_1} \ldots,^{A_{j-1}} S_j \rangle \rightsquigarrow L_1 \otimes \ldots \otimes L_n$ and $P, \langle S_p,^{A_k^{-1}} \ldots,^{A_1^{-1}} S_q \rangle, S_q \vdash_c A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ are proven to be sound and complete w.r.t. ($M, \langle S_1,^{A_1} \ldots,^{A_{j-1}} S_j \rangle \models_p L_1 \otimes \ldots \otimes L_n$ and $M, \langle S_1,^{A_1} \ldots,^{A_{j-1}} S_j \rangle \not\models_c L_1 \otimes \ldots \otimes L_n$) and $M, \langle S_p,^{A_k^{-1}} \ldots,^{A_1^{-1}} S_q \rangle, S_q \vdash_c A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ respectively. □

Lemma 15 (Soundness of $\vdash$ Axiom)**.** *Let P be a serial-Horn program, ϕ a serial-Horn goal, and π be a path.*

$$\textit{If } P, \pi \vdash () \textit{ then } P, \pi \models ()$$

Proof. $P, \pi \vdash ()$ holds for every 1-path π. Since the empty transaction $()$ belongs to any interpretation of a 1-path, then for every M', $M', \pi \models ()$. Thus, in particular for every model M of P, it holds that $M', \pi \models ()$ and consequently by definition 20 it is true that $P, \pi \models ()$ □

Lemma 16 (Soundness of $\vdash$ Rules)**.** *Let P be a serial-Horn program, $\phi_1 \otimes \ldots \otimes \phi_k$ be a completely unfolded serial-Horn goal and π be a path. Then the following conditions are true:*

1. *Assume that $\phi_1 \leftarrow \psi$ is a rule in P.*
   If $P, \pi \models \psi \otimes \ldots \otimes \phi_k$ then $P, \pi \models \phi_1 \otimes \ldots \otimes \phi_k$
2. *Assume that $\mathcal{O}^d(D_1) \models \phi_1$*
   If $P, \langle (D_1,E_1),^{A_1} \ldots,^{A_{f-1}} (D_f,E_f) \rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ then
   $P, \langle (D_1,E_1),^{A_1} \ldots,^{A_{f-1}} (D_f,E_f) \rangle \models \phi_1 \otimes \ldots \otimes \phi_k$
3. *Assume that $\mathcal{O}^t(D_0,D_1) \models \phi_1$*
   If $P, \langle (D_1,E_1),^{A_1} \ldots,^{A_{f-1}} (D_f,E_f) \rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ then
   $P, \langle ((D_0,E_1),^{\phi_1} D_1,E_1),^{A_1} \ldots,^{A_{f-1}} (D_f,E_f) \rangle \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$
4. *Assume that $\mathcal{O}^e(E_0,E_1) \models \phi_1$*
   If $P, \langle (D_1,E_1),^{A_1} \ldots,^{A_{f-1}} (D_f,E_f) \rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ then
   $P, \langle ((D_1,E_0),^{\phi_1} D_1,E_1),^{A_1} \ldots,^{A_{f-1}} (D_f,E_f) \rangle \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$
5. *Assume that*
   - $P, \langle S_1,^{A_1} \ldots,^{A_{j-1}} S_j \rangle \rightsquigarrow L_1 \otimes \ldots \otimes L_n$
   - $S_1,^{A_1} \ldots,^{A_{p-1}} S_p$ *is the rollback path of the path* $S_1,^{A_1} \ldots,^{A_{j-1}} S_j$ *(cf. Definition 15)*
   - $\texttt{Inv}(\texttt{Seq}(\langle S_1,^{A_1} \ldots,^{A_{p-1}} S_p \rangle)) = A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ *(cf. Definition 16)*
   - $P, \langle S_p,^{A_k^{-1}} \ldots,^{A_1^{-1}} S_q \rangle, S_q \vdash_c A_k^{-1} \otimes \ldots \otimes A_1^{-1}$

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

Proof. We prove each item in turn.

1. Not applicable since we are talking only about unfolded formulas

2. Assume that $\mathcal{O}^d(D_1) \models \phi_1$ and that the following $P, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ holds. Then we need to prove that $P, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$
   By definition 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 we know that $P, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$ then we also know that $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$. Then by the serial conjunction case we can conclude that $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$ for every M that models P. Consequently as expected, it holds that $P, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$
3. Assume that $\mathcal{O}^t(D_0, D_1) \models \phi_1$ and that the following $P, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ holds. Then we need to prove that $P, \langle ((D_0, E_1),^{\phi_1} D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$
   By definition we know that for every M (and in particular, for every M that models P), $M, \langle (D_0, E_1),^{\phi_1} (D_1, E_1)\rangle \models \phi_1$. Since we know that $P, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$ then we know $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$. Then by the serial conjunction case we can conclude that $M, \langle (D_0, E_1),^{\phi_1} (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$ for every M that models P. Consequently, as intended, it holds that $P, \langle (D_0, E_1),^{\phi_1} (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$
4. Assume that $\mathcal{O}^e(E_0, E_1) \models \phi_1$ and that the following $P, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ holds. Then we need to prove that $P, \langle ((D_0, E_1),^{\phi_1} D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$
   By definition we know that for every M (and in particular, for every M that models P), $M, \langle (D_1, E_0),^{\phi_1} (D_1, E_0)\rangle \models \phi_1$. Since we know $P, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$ then we know that $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$. Then by the serial conjunction case we can conclude that $M, \langle (D_1, E_0),^{\phi_1} (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$ for every M that models P. Consequently as intended it holds that $P, \langle (D_1, E_0),^{\phi_1} (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$
5. Assume the following:
   - $P, \langle S_1,^{A_1} \ldots,^{A_{j-1}} S_j\rangle \rightsquigarrow L_1 \otimes \ldots \otimes L_n$
   - $S_1,^{A_1} \ldots,^{A_{p-1}} S_p$ is the rollback path of $S_1,^{A_1} \ldots,^{A_{j-1}} S_j$ (cf. Definition 15)
   - $\mathtt{Inv}(\mathtt{Seq}(\langle S_1,^{A_1} \ldots,^{A_{p-1}} S_p\rangle)) = A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ (cf. Definition 16)
   - $P, \langle S_p,^{A_k^{-1}} \ldots,^{A_1^{-1}} S_q\rangle, S_q \vdash_c A_k^{-1} \otimes \ldots \otimes A_1^{-1}$

   We want to prove that if $P, \langle S_q,^{A_q} \ldots,^{A_{f-1}} S_f\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$ then $P, \langle S_1,^{A_1} \ldots,^{A_{p-1}} S_p,^{A_{p-1}^{-1}} \ldots,^{A_1^{-1}} S_q,^{A_q} \ldots,^{A_{f-1}} S_f\rangle \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$
   Since we have $P, \langle S_q,^{A_q} \ldots,^{A_{f-1}} S_f\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$ we know that for every model M of P that $M, \langle S_q,^{A_q} \ldots,^{A_{f-1}} S_f\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$. Moreover, since all conditions of point 5 apply, by Lemma 21 we have that for every model M of P $M, \langle S_1,^{A_1} \ldots,^{A_{p-1}} S_p,^{A_{p-1}^{-1}} \ldots,^{A_1^{-1}} S_q\rangle \rightsquigarrow \phi_1 \otimes \ldots \otimes \phi_k$. Moreover, since we know that $M, \langle S_q,^{A_q} \ldots,^{A_{f-1}} S_f\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$, we also have $M, \langle S_1,^{A_1} \ldots,^{A_{p-1}} S_p,^{A_{p-1}^{-1}} \ldots,^{A_1^{-1}} S_q,^{A_q} \ldots,^{A_{f-1}} S_f\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$ and thus by definition of the executional entailment it holds $P, \langle S_1,^{A_1} \ldots,^{A_{p-1}} S_p,^{A_{p-1}^{-1}} \ldots,^{A_1^{-1}} S_q,^{A_q} \ldots,^{A_{f-1}} S_f\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$.

□

Lemma 17 (Completeness of $\vdash$)**.** *Let P be a serial-Horn program, ϕ be a completely unfolded serial-Horn goal, and π be a path.*

$$\textit{If } P, \pi \models \phi \textit{ then } P, \pi \vdash \phi$$

Proof. We prove by induction on the size of ϕ.
Base Case k = 0:
If $P, \pi \models ()$ then $P, \pi \vdash ()$
$P, \pi \models ()$ implies that for every model M of P, $M, \pi \models ()$. Since $()$ is the empty transaction satisfied in paths of size 1, $M, \pi \models ()$ only if π is a 1-path. If this is the case, then for any sequents of size 1 (that correspond to 1-paths), the sequence is complete and $P, \pi \vdash ()$.

Inductive Case k = j+1:
Let's assume that this sentence is true for values for $\phi = \phi_1 \otimes \ldots \otimes \phi_j$. We prove that it remains true for $\phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$. So we have that $P, \pi \models \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ and want to prove that $P, \pi \vdash \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$. Since $P, \pi \models \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ we know that for every model M of P, $M, \pi \models \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$,

and consequently, there are two possible cases for this to be true:

1. *Serial Conjunction Case*.
   Then, it exists a split $\pi_1 \circ \pi_2$ of π s.t. $M, \pi_1 \models \phi_1 \otimes \ldots \otimes \phi_j$ and $M, \pi_2 \models \phi_{j+1}$. Consequently, since this is true for all models M we know that $P, \pi_1 \vdash \phi_1 \otimes \ldots \otimes \phi_j$. Since ϕ is completely unfolded, then ϕ_{j+1} is an atomic action defined in the oracles. Then $M, \pi_2 \models \phi_{j+1}$ iff one of the following cases is true:

   (a) $\pi_2 = \langle D, E \rangle$ and $\mathcal{O}^d(D) \models \phi_{j+1}$. If this is the case then $P, (D, E) \vdash \phi_{j+1}$
   (b) $\pi_2 = \langle (D_1, E),^{\phi_{j+1}} (D_2, E) \rangle$ and $\mathcal{O}^t(D_1, D_2) \models \phi_{j+1}$. If this is the case then by definition $P, (D_1, E),^{\phi_{j+1}} (D_2, E) \vdash \phi_{j+1}$.
   (c) $\pi_2 = \langle (D, E_1),^{\phi_{j+1}} (D, E_2) \rangle$ and $\mathcal{O}^e(E_1, E_2) \models \phi_{j+1}$. If this is the case then by definition $P, (D, E_1),^{\phi_{j+1}} (D, E_2) \vdash \phi_{j+1}$.

   Since in all cases, $P, \pi_2 \vdash \phi_{j+1}$, then we can conclude that $P, \pi \vdash \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$
2. *Compensating Case*.
   Then it exists a split $\pi_1 \circ \pi_2$ of π s.t. $M, \pi_1 \leadsto \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ and $M, \pi_2 \models \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$. By Lemma 21 we know that $M, \pi_1 \leadsto \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ is complete w.r.t. to rule 5 of the proof theory definition. Thus, we known that if $P, \pi_2 \vdash \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$, then $P, \pi \vdash \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$. So it remains to show that $P, \pi_2 \vdash \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ However, it is easy to see that it exists a $\pi_2' \circ \pi_2'' = \pi_2$ s.t. $M, \pi_2'' \models_c \phi_{j+1}$ and $M, \pi_2' \models \phi_1 \otimes \ldots \otimes \phi_j$ Then by Lemma 20 we know that $M, \pi_2'' \models_c \phi_{j+1}$ implies $P, \pi_2'' \vdash_c \phi_{j+1}$ which implies $P, \pi_2'' \vdash \phi_{j+1}$. From applying the assumption of the inductive case we have that $P, \pi' \vdash \phi_1 \otimes \ldots \otimes \phi_j$ and thus $P, \pi_2 \vdash \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$

□

Lemma 18. *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 $M, \pi_2 \not\models_c \phi$ then $M, \pi \not\models \psi \otimes \phi$*

Proof. Assume $M, \pi \models_c \psi \otimes \phi$ then $\exists$ a split π', π'' s.t. $M, \pi' \models_c \psi$ and $M, \pi'' \models_c \phi$. Since ψ is an atomic action then such split must be $\pi_1 \circ \pi_2$ and thus since $M, \pi_2 \not\models_c \phi$ then the assumption fails. □

Lemma 19. *Let P be a serial-Horn program, M be a model of P, π be a path, ϕ a serial goal. Then:*

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

Proof. **Soundness of Axiom**:
If $\phi = ()$ then for any 1-path π, $P, \pi \vdash_c ()$. Moreover, since $()$ represents the empty transaction which is tautologically true in any path of length 1.
Soundness of Rules:
We prove each rule in turn:

1. Assume it exists a rule $L_1 \leftarrow B_1 \otimes \ldots \otimes B_j$ in P. Then we want to prove that if $M, \pi \models_c B_1 \otimes \ldots \otimes B_j \otimes L_2 \otimes \ldots \otimes L_k$ then $M, \pi \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$
   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 it exists 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, then by Lemma 3 we have that $M, \pi_1 \models_c L_1$ and thus $M, \pi \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$
2. Assume that $\mathcal{O}^d(D_1) \models L_1$. We want to prove that if $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_f} (D_f, E_f) \rangle \models_c L_2 \otimes \ldots \otimes L_k$ then it also holds that $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_f} (D_f, E_f) \rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$
   Since $\mathcal{O}^d(D_1) \models L_1$, then 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 $\models_c$ we can conclude that
   $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_f} (D_f, E_f) \rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$
3. Assume that $\mathcal{O}^t(D_0, D_1) \models L_1$. We want to prove that if $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_f} (D_f, E_f) \rangle \models_c L_2 \otimes \ldots \otimes L_k$ then it also holds that $M, \langle (D_0, E_1),^{L_1} (D_1, E_1),^{A_1} \ldots,^{A_f} (D_f, E_f) \rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$
   Since $\mathcal{O}^t(D_0, D_1) \models L_1$, then we know that for every interpretation, $M, \langle (D_0, E_1),^{L_1} (D_1, E_1) \rangle \models_c L_1$. Thus, by definition of the serial conjunction case of $\models_c$ we can conclude that
   $M, \langle (D_0, E_1),^{L_1} (D_1, E_1),^{A_1} \ldots,^{A_f} (D_f, E_f) \rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$
4. Assume that $\mathcal{O}^e(E_0, E_1) \models L_1$. We want to prove that if $M, \langle (D_1, E_1),^{A_1} \ldots,^{A_f} (D_f, E_f) \rangle \models_c L_2 \otimes \ldots \otimes L_k$ then it also holds that $M, \langle (D_1, E_0),^{L_1} (D_1, E_1),^{A_1} \ldots,^{A_f} (D_f, E_f) \rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$
   Since $\mathcal{O}^e(E_0, E_1) \models L_1$, then we know that for every interpretation, $M, \langle (D_1, E_0),^{L_1} (D_1, E_1) \rangle \models_c L_1$. Thus, by definition of the serial conjunc-

tion case of $\models_c$ we can conclude that $M, \langle (D_1, E_0),^{L_1} (D_1, E_1),^{A_1} \ldots,^{A_f} (D_f, E_f)\rangle \models_c L_1 \otimes L_2 \otimes \ldots \otimes L_k$

□

Lemma 20. *Let P be a serial-Horn program, M be a model of P, π be a path, ϕ a completely unfolded serial-goal. Then the following is true:*

If $M, \pi \models_c \phi$ then $P, \pi \vdash_c \phi$

Proof. Proof by induction on the length of $\phi = \phi_1 \otimes \ldots \otimes \phi_k$.

Base Case k = 0

If $k = 0$ then $\phi = ()$. Thus $M, \pi \models_c ()$ for any 1-path π. By definition, $P, \pi \models_c ()$ also holds and it is complete.

Inductive Case k = j+1

We assume that the statement is true for all values up to j and prove that it still holds for values $j + 1$. We start by assuming that $M, \pi \models_c \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ is true. Then we know by the serial conjunction case of $\models_c$ that it exists a split $\pi_1 \circ \pi_2$ s.t. $M, \pi_1 \models_c \phi_1 \otimes \ldots \otimes \phi_j$ and $M, \pi_2 \models \phi_{j+1}$. Now since ϕ is a completely unfolded formula are three possible cases for ϕ_{j+1}:

1. $\pi_2 = \langle (D, E)\rangle \wedge \mathcal{O}^d(D) \models \phi_{j+1}$ then for any state E, by rule 2. of Definition 21 we have that $P, \langle (D, E)\rangle \vdash_c \phi_{j+1}$
2. $\pi_2 = \langle (D, E),^{\phi_{j+1}} (D_1, E)\rangle \wedge \mathcal{O}^t(D, D_1) \models \phi_{j+1}$ then for any state E, by rule 2. of Definition 21 we have that $P, \langle (D, E),^{\phi_{j+1}} (D_1, E)\rangle \vdash_c \phi_{j+1}$
3. $\pi_2 = \langle (D, E),^{\phi_{j+1}} (D, E_1)\rangle \wedge \mathcal{O}^e(E, E_1) \models \phi_{j+1}$ then for any state E, by rule 2. of Definition 21 we have that $P, \langle (D, E),^{\phi_{j+1}} (D, E_1)\rangle \vdash_c \phi_{j+1}$

Moreover, since $P, \pi_1 \vdash_c \phi_1 \otimes \ldots \otimes \phi_j$ and $P, \pi_2 \vdash_c \phi_{j+1}$ then since ϕ_2 starts in the last state of π_1 and $P, \pi_2 \vdash_c \phi_{j+1}$ is complete, then $P, \pi_1 \circ \pi_2 \vdash_c \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ □

Lemma 21 (Soundness & Completeness rule-5. w.r.t. $\rightsquigarrow$)**.** *Let P be a serial-Horn program, $L_1 \otimes \ldots \otimes L_k$ be a completely unfolded serial-Horn goal. Let the following three conditions be true:*

- $P, \langle S_1,^{A_1} \ldots,^{A_{j-1}} S_j\rangle \rightsquigarrow L_1 \otimes \ldots \otimes L_n$
- $S_1,^{A_1} \ldots,^{A_{p-1}} S_p$ *is the rollback path of the path* $S_1,^{A_1} \ldots,^{A_{j-1}} S_j$ *(cf. Definition 15)*
- $\texttt{Inv}(\texttt{Seq}(\langle S_1,^{A_1} \ldots,^{A_{p-1}} S_p\rangle)) = A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ *(cf. Definition 16)*
- $P, \langle S_p,^{A_k^{-1}} \ldots,^{A_1^{-1}} S_q\rangle, S_q \vdash_c A_k^{-1} \otimes \ldots \otimes A_1^{-1}$

Then, for every model M of P, it holds that: $M, \langle S_1,^{A_1} \ldots,^{A_{p-1}} S_p,^{A_{p-1}^{-1}} \ldots,^{A_1^{-1}} S_q\rangle \rightsquigarrow L_1 \otimes \ldots \otimes L_k$

Proof. Immediate since $P, \langle S_1,^{A_1} \ldots,^{A_{j-1}} S_j\rangle \rightsquigarrow L_1 \otimes \ldots \otimes L_n$ and $P, \langle S_p,^{A_k^{-1}} \ldots,^{A_1^{-1}} S_q\rangle, S_q \vdash_c A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ are proven to be sound and complete w.r.t. ($M, \langle S_1,^{A_1} \ldots,^{A_{j-1}} S_j\rangle \models_p L_1 \otimes \ldots \otimes L_n$ and $M, \langle S_1,^{A_1} \ldots,^{A_{j-1}} S_j\rangle \not\models_c L_1 \otimes \ldots \otimes L_n$) and $M, \langle S_p,^{A_k^{-1}} \ldots,^{A_1^{-1}} S_q\rangle, S_q \vdash_c A_k^{-1} \otimes \ldots \otimes A_1^{-1}$ respectively. □

Lemma 22 (Soundness of $\vdash$ Axiom)**.** *Let P be a serial-Horn program, ϕ a serial-Horn goal, and π be a path.*

If $P, \pi \vdash ()$ then $P, \pi \models ()$

Proof. $P, \pi \vdash ()$ holds for every 1-path π. Since the empty transaction $()$ belongs to any interpretation of a 1-path, then for every M', $M', \pi \models ()$. Thus, in particular for every model M of P, it holds that $M', \pi \models ()$ and consequently by definition 20 it is true that $P, \pi \models ()$ □

Lemma 23 (Soundness of $\vdash$ Rules)**.** *Let P be a serial-Horn program, $\phi_1 \otimes \ldots \otimes \phi_k$ be a completely unfolded serial-Horn goal and π be a path. Then the following conditions are true:*

1. *Assume that $\phi_1 \leftarrow \psi$ is a rule in P. If $P, \pi \models \psi \otimes \ldots \otimes \phi_k$ then $P, \pi \models \phi_1 \otimes \ldots \otimes \phi_k$*
2. *Assume that $\mathcal{O}^d(D_1) \models \phi_1$.*
   If $P, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ then
   $P, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \ldots \otimes \phi_k$
3. *Assume that $\mathcal{O}^t(D_0, D_1) \models \phi_1$.*
   If $P, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ then
   $P, \langle ((D_0, E_1),^{\phi_1} D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$
4. *Assume that $\mathcal{O}^e(E_0, E_1) \models \phi_1$.*
   If $P, \langle (D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_2 \otimes \ldots \otimes \phi_k$ then
   $P, \langle ((D_1, E_0),^{\phi_1} D_1, E_1),^{A_1} \ldots,^{A_{f-1}} (D_f, E_f)\rangle \models \phi_1 \otimes \phi_2 \otimes \ldots \otimes \phi_k$
5. *Assume that*

   - $P, \langle S_1,^{A_1} \ldots,^{A_{j-1}} S_j\rangle \rightsquigarrow L_1 \otimes \ldots \otimes L_n$

- $S_1,^{A_1}\ldots,^{A_{p-1}} S_p$ *is the rollback path of the path* $S_1,^{A_1}\ldots,^{A_{j-1}} S_j$ *(cf. Definition 15)*
- $\texttt{Inv}(\texttt{Seq}(\langle S_1,^{A_1}\ldots,^{A_{p-1}} S_p\rangle)) = A_k^{-1}\otimes\ldots\otimes A_1^{-1}$ *(cf. Definition 16)*
- $P,\langle S_p,^{A_k^{-1}}\ldots,^{A_1^{-1}} S_q\rangle, S_q \vdash_c A_k^{-1}\otimes\ldots\otimes A_1^{-1}$

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

Proof. We prove each item in turn.

1. Not applicable since we are talking only about unfolded formulas
2. Assume that $\mathcal{O}^d(D_1)\models\phi_1$.
   If $P,\langle(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_2\otimes\ldots\otimes\phi_k$ then we need to prove $P,\langle(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_1\otimes\ldots\otimes\phi_k$
   By definition 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 we know that $P,\langle(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_1\otimes\ldots\otimes\phi_k$ then we also know that $M,\langle(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_2\otimes\ldots\otimes\phi_k$. Then by the serial conjunction case we can conclude that $M,\langle(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_1\otimes\ldots\otimes\phi_k$ for every M that models P. Consequently, as intended, it holds that $P,\langle(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_1\otimes\ldots\otimes\phi_k$
3. Assume that $\mathcal{O}^t(D_0,D_1)\models\phi_1$.
   If $P,\langle(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_2\otimes\ldots\otimes\phi_k$ then we need to prove that it holds $P,\langle((D_0,E_1),^{\phi_1}D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_1\otimes\phi_2\otimes\ldots\otimes\phi_k$
   By definition we know that for every M (and in particular, for every M that models P), $M,\langle(D_0,E_1),^{\phi_1}(D_1,E_1)\rangle\models\phi_1$. Since we know $P,\langle(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_1\otimes\ldots\otimes\phi_k$ then we know that $M,\langle(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_2\otimes\ldots\otimes\phi_k$. Then by the serial conjunction case we can conclude that $M,\langle(D_0,E_1),^{\phi_1}(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_1\otimes\ldots\otimes\phi_k$ for every M that models P. Consequently, as intended, it holds that $P,\langle(D_0,E_1),^{\phi_1}(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_1\otimes\ldots\otimes\phi_k$
4. Assume that $\mathcal{O}^e(E_0,E_1)\models\phi_1$.
   If $P,\langle(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_2\otimes\ldots\otimes\phi_k$ then we need to prove that it holds $P,\langle((D_0,E_1),^{\phi_1}D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_1\otimes\phi_2\otimes\ldots\otimes\phi_k$
   By definition we know that for every M (and in particular, for every M that models P), $M,\langle(D_1,E_0),^{\phi_1}(D_1,E_0)\rangle\models\phi_1$. Since we know that $P,\langle(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_1\otimes\ldots\otimes\phi_k$ then we also know that $M,\langle(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_2\otimes\ldots\otimes\phi_k$. Then by the serial conjunction case we can conclude that $M,\langle(D_1,E_0),^{\phi_1}(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_1\otimes\ldots\otimes\phi_k$ for every M that models P. Consequently it holds that $P,\langle(D_1,E_0),^{\phi_1}(D_1,E_1),^{A_1}\ldots,^{A_{f-1}}(D_f,E_f)\rangle\models\phi_1\otimes\ldots\otimes\phi_k$
5. Assume that
   - $P,\langle S_1,^{A_1}\ldots,^{A_{j-1}} S_j\rangle \rightsquigarrow L_1\otimes\ldots\otimes L_n$
   - $S_1,^{A_1}\ldots,^{A_{p-1}} S_p$ is the rollback path of $S_1,^{A_1}\ldots,^{A_{j-1}} S_j$ (cf. Definition 15)
   - $\texttt{Inv}(\texttt{Seq}(\langle S_1,^{A_1}\ldots,^{A_{p-1}} S_p\rangle)) = A_k^{-1}\otimes\ldots\otimes A_1^{-1}$ (cf. Definition 16)
   - $P,\langle S_p,^{A_k^{-1}}\ldots,^{A_1^{-1}} S_q\rangle, S_q\vdash_c A_k^{-1}\otimes\ldots\otimes A_1^{-1}$

   If $P,\langle S_q,^{A_q}\ldots,^{A_{f-1}} S_f\rangle\models\phi_1\otimes\ldots\otimes\phi_k$ then we need to prove that $P,\langle S_1,^{A_1}\ldots,^{A_{p-1}} S_p,^{A_{p-1}^{-1}}\ldots,^{A_1^{-1}} S_q,^{A_q}\ldots,^{A_{f-1}} S_f\rangle\models\phi_1\otimes\phi_2\otimes\ldots\otimes\phi_k$
   Since we have $P,\langle S_q,^{A_q}\ldots,^{A_{f-1}} S_f\rangle\models\phi_1\otimes\ldots\otimes\phi_k$ we know that for every model M of P that $M,\langle S_q,^{A_q}\ldots,^{A_{f-1}} S_f\rangle\models\phi_1\otimes\ldots\otimes\phi_k$. Moreover, since all conditions of point 5 apply, by Lemma 21 we know for every model M of P: $M,\langle S_1,^{A_1}\ldots,^{A_{p-1}} S_p,^{A_{p-1}^{-1}}\ldots,^{A_1^{-1}} S_q\rangle\rightsquigarrow\phi_1\otimes\ldots\otimes\phi_k$. Additionally, since we have that $M,\langle S_q,^{A_q}\ldots,^{A_{f-1}} S_f\rangle\models\phi_1\otimes\ldots\otimes\phi_k$, then we know $M,\langle S_1,^{A_1}\ldots,^{A_{p-1}} S_p,^{A_{p-1}^{-1}}\ldots,^{A_1^{-1}} S_q,^{A_q}\ldots,^{A_{f-1}} S_f\rangle\models\phi_1\otimes\ldots\otimes\phi_k$ and thus by definition of the executional entailment it holds $P,\langle S_1,^{A_1}\ldots,^{A_{p-1}} S_p,^{A_{p-1}^{-1}}\ldots,^{A_1^{-1}} S_q,^{A_q}\ldots,^{A_{f-1}} S_f\rangle\models\phi_1\otimes\ldots\otimes\phi_k$.

□

Lemma 24 (Completeness of $\vdash$). *Let P be a serial-Horn program, ϕ be a completely unfolded serial-Horn goal, and π be a path.*

$$\textit{If } P,\pi\models\phi \textit{ then } P,\pi\vdash\phi$$

Proof. We prove by induction on the size of ϕ.
Base Case k = 0:
If $P,\pi\models()$ then $P,\pi\vdash()$
$P,\pi\models()$ implies that for every model M of P, $M,\pi\models()$. Since $()$ is the empty transaction satisfied

in paths of size 1, $M, \pi \models ()$ only if π is a 1-path. If this is the case, then for any sequents of size 1 (that correspond to 1-paths), the sequence is complete and $P, \pi \vdash ()$.

Inductive Case k = j+1:
Let's assume that this sentence is true for values for $\phi = \phi_1 \otimes \ldots \otimes \phi_j$. We prove that it remains true for $\phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$. So we have that $P, \pi \models \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ and want to prove that $P, \pi \vdash \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$. Since $P, \pi \models \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ we know that for every model M of P, $M, \pi \models \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$, and consequently, there are two possible cases for this to be true:

1. *Serial Conjunction Case*.
   Then, it exists a split $\pi_1 \circ \pi_2$ of π s.t. $M, \pi_1 \models \phi_1 \otimes \ldots \otimes \phi_j$ and $M, \pi_2 \models \phi_{j+1}$. Consequently, since this is true for all models M we know that $P, \pi_1 \vdash \phi_1 \otimes \ldots \otimes \phi_j$. Since ϕ is completely unfolded, then ϕ_{j+1} is an atomic action defined in the oracles. Then $M, \pi_2 \models \phi_{j+1}$ iff one of the following cases is true:
   (a) $\pi_2 = \langle D, E\rangle$ and $\mathcal{O}^d(D) \models \phi_{j+1}$. If this is the case then $P, (D, E) \vdash \phi_{j+1}$
   (b) $\pi_2 = \langle (D_1, E),^{\phi_{j+1}} (D_2, E)\rangle$ and $\mathcal{O}^t(D_1, D_2) \models \phi_{j+1}$. If this is the case then by definition $P, (D_1, E),^{\phi_{j+1}} (D_2, E) \vdash \phi_{j+1}$.
   (c) $\pi_2 = \langle (D, E_1),^{\phi_{j+1}} (D, E_2)\rangle$ and $\mathcal{O}^e(E_1, E_2) \models \phi_{j+1}$. If this is the case then by definition $P, (D, E_1),^{\phi_{j+1}} (D, E_2) \vdash \phi_{j+1}$.

   Since in all cases, $P, \pi_2 \vdash \phi_{j+1}$, then we can conclude that $P, \pi \vdash \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$
2. *Compensating Case*.
   Then it exists a split $\pi_1 \circ \pi_2$ of π s.t. $M, \pi_1 \rightsquigarrow \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ and $M, \pi_2 \models \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$. By Lemma 21 we know that $M, \pi_1 \rightsquigarrow \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ is complete w.r.t. to rule 5 of the proof theory definition. Thus, we known that if $P, \pi_2 \vdash \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$, then $P, \pi \vdash \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$. So it remains to show that $P, \pi_2 \vdash \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$ However, it is easy to see that it exists a $\pi_2' \circ \pi_2'' = \pi_2$ s.t. $M, \pi_2'' \models_c \phi_{j+1}$ and $M, \pi_2' \models \phi_1 \otimes \ldots \otimes \phi_j$ Then by Lemma 20 we know that $M, \pi_2'' \models_c \phi_{j+1}$ implies $P, \pi_2'' \vdash_c \phi_{j+1}$ which implies $P, \pi_2'' \vdash \phi_{j+1}$. From applying the assumption of the inductive case we have that $P, \pi' \vdash \phi_1 \otimes \ldots \otimes \phi_j$ and thus $P, \pi_2 \vdash \phi_1 \otimes \ldots \otimes \phi_j \otimes \phi_{j+1}$

□

Proofs of the enunciated results

Proof of Proposition 1. We prove each item in turn.

1. If $M, \pi \models_p \phi$ and $M, \pi \not\models_c \phi$ then $\exists a$ s.t. $M, \pi_{\texttt{end}} \models_p a$ 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 this statement holds by Case 1. of the Definitions 14 and 13.
   Induction Step:
   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. $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$. Then we know that $M, \pi \not\models_p \phi$ and $M, \pi \models_c \neg\phi$ which is always false. Since the antecedent of the implication is false, the statement is trivially satisfied.
   Conjunction: Let's 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. $M, \pi_{\texttt{end}} \models_p a$ 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. $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$. Then we have two 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 hypothesis and thus conclude that $\exists a$ s.t. $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$; 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 latter case we can then apply apply the hypothesis and thus conclude that $\exists a$ s.t. $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$;
   Disjunction: Let's 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. $M, \pi_{\texttt{end}} \models_p a$ 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. $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$. Then we have two cases:
   1) Suppose that $M, \pi \models_p \phi$ and $M, \pi \not\models_p \psi$ and $M, \pi \not\models_c \phi$ hold, then we can apply apply the hypothesis and thus conclude that $\exists a$ s.t. $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$; 2) Suppose that $M, \pi \models_p \psi$ and $M, \pi \not\models_p \psi$ and $M, \pi \not\models_c \phi$ hold, then we can apply apply the hypothesis and thus conclude that $\exists a$ s.t. $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$;
   Serial Conjunction: Let's 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. $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$.
This is equivalent to If ([(1) $\exists \pi_1, \pi_2 : \pi_1 \circ \pi_2 = \pi$ and $M, \pi_1 \models_c \phi$ and $M, \pi_2 \models_p \psi$] or [(2) $M, \pi \models_p \phi$ and $M, \pi \not\models_c \phi$] and $\forall \pi_3, \pi_4 : \pi_3 \circ \pi_4 = \pi \rightarrow M, \pi_3 \not\models_c \phi$ or $M, \pi_4 \not\models_c \psi$) then $\exists a$ s.t. $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$.
Suppose (2) is the case. Then since by hypothesis the result holds for ϕ, (2) is enough to conclude that $\exists a$ s.t. $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$. Suppose (1) is the case and $\forall \pi_3, \pi_4 : \pi_3 \circ \pi_4 = \pi$ either $M, \pi_3 \not\models_c \phi$ or $M, \pi_4 \not\models_c \psi$ (since $M, \pi \not\models_c \phi \otimes \psi$). Since this latter statement holds for every split of π, then it also holds for $\pi = \pi_1 \circ \pi_2$. However, by (1) we know that $M, \pi_1 \models_c \phi$ and thus $M, \pi_2 \not\models_c \psi$ must hold. Since $M, \pi_2 \models_p \psi$ and $M, \pi_2 \not\models_c \psi$ hold, by hypothesis we have that $\exists a$ s.t. $M, \pi_{\texttt{end}} \models_p a$ and $M, \pi_{\texttt{end}} \not\models_c a$ for path π_2. Since π is composed by $\pi_1 \circ \pi_2$ then this result holds for $\phi \otimes \psi$.
□

2. If $M, \pi \models_p \phi$ and $M, \pi \not\models_c \phi$ then $M, \pi \models_p \phi \otimes \psi$
This comes immediately by the point a) of the serial conjunction of Definition 14
3. If $M, \pi \models_c \phi$ then $M, \pi \models_p \phi$
We prove by induction on the structure of ϕ:
Base Case: If ϕ is an atom, then this statement holds immediately by Case 1. of the Definitions 14 and 13.
Induction Step:
Negation: Let's assume that the result holds for ϕ. We want to prove that if $M, \pi \models_c \neg\phi$ then $M, \pi \models_p \neg\phi$. Assume that $M, \pi \models_c \neg\phi$ then $\phi \notin M(\pi)$
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 $M, \pi \models_p \phi \wedge \psi$. Assume that $M, \pi \models_c \phi \wedge \psi$. Then $M, \pi \models_c \phi$ and $M, \pi \models_c \psi$. Applying the induction hypothesis, then 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 $M, \pi \models_p \phi \vee \psi$. If $M, \pi \models_c \phi \vee \psi$ then either $M, \pi \models_c \phi$ or $M, \pi \models_c \psi$. So 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 $M, \pi \models_p \phi \vee \psi$ 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 $M, \pi \models_p \phi \vee \psi$ for any transaction ϕ.
Serial Conjunction: Let's 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 it 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 $M, \pi_2 \models_p \psi$. Then we are in conditions of applying point b) of the serial conjunction case of the partial satisfaction, and thus $M, \pi \models_p \phi_1 \otimes \phi_2$.
4. $M, \pi \models_c \phi_P$ iff $M, \pi \models_p \phi_P$
This statement holds immediately by Case 1. of the Definitions 14 and 13.

□

Proof of Theorem 2. We prove each item in turn,

1. If $M, \pi \models_c \phi$ then $M, \pi \models \phi$
We prove by induction on the structure of ϕ:
Base Case: If ϕ is an atom, then this statement holds by Case 1. of the Definitions 18 and 13.
Induction Step:
Negation: Let's assume that this result holds for ϕ. Then we want to prove that $M, \pi \models_c \neg\phi$ implies $M, \pi \models \neg\phi$, i.e. if $\phi \notin M(\pi)$ then it is also not the case that $M, \pi \models \phi$. Let's assume that $M, \pi \models \phi$ is true and prove it leads to a contradiction. Since $\phi \notin M(\pi)$ then the only case where ϕ can succeed in $\models$ is if it exists a split $\pi_1 \circ \pi_2 of \pi$ s.t. $M, \pi_1 \rightsquigarrow \phi$ and $M, \pi_2 \models \phi$. We will prove that $M, \pi_1 \rightsquigarrow \phi$ is impossible. Since ϕ is an atom, then for any path π' $M, \pi' \models_p \phi$ and $M, \pi' \not\models_c \phi$ only if π' is a 1-path. Then the rollback path of $\pi' = \pi'$ and then $\texttt{Seq}(\pi') = \emptyset$. Consequently, for any π_1 the conditions to apply $M, \pi_1 \rightsquigarrow \phi$ do not hold and thus this last statement must be false. As a result, it is not the case that $M, \pi \models \phi$ and thus $M, \pi \models \neg\phi$ as desired. *Disjunction*: Let's assume that this 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 $M, \pi \models_c \phi$ (1) or $M, \pi \models_c \phi \vee \psi$ (2). Let's assume that (1) is true. Then by hypothesis we know that $M, \pi \models \phi$ and thus by definition $M, \pi \models \phi \vee \psi$. Furthermore, let's assume that (2) is true. Then by hy-

pothesis we know that $M, \pi \models \psi$ and thus 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 one of the following holds $M, \pi \models_c \phi$ and $M, \pi \models_c \psi$. Then by induction hypothesis we can conclude that $M, \pi \models \phi$ and $M, \pi \models \psi$ hold and thus by definition, $M, \pi \models \phi \wedge \psi$.
Serial Conjunction: Let's assume that this 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 it 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 know that $M, \pi_1 \models \phi$ and $M, \pi_2 \models \psi$. Since $\pi_1 \circ \pi_2$ are splits of π, then by the serial conjunction case of definition 18 we know that $M, \pi \models \phi \otimes \psi$.

2. $M, \pi \models_c \phi$ iff $M, \pi \models \phi$ where π is a path without external actions in the annotated transitions.
   If π does not contain annotations for external actions, then ϕ can never be an external action or contain positive formulas of external actions by definition of interpretations of formulas. Moreover, by Lemma 2 we know that if π does not contain annotations of external actions then it is impossible to construct a consistency preserving path, i.e. since π does not contain annotations of external actions, then for any subpath π' of π $M, \pi' \rightsquigarrow \phi$ does not hold. Thus $M, \pi \models \phi$ iff $M, \pi' \models \phi$.

$\square$

Proof of Theorem 3. $\Rightarrow$:
We want to prove that if $P, \pi' \models_{TR} \phi$ then $P, \pi \models_{ETR} \phi$. Since $P, \pi' \models_{TR} \phi$ we know that for every model M of P (in $\mathcal{TR}$) then $M, \pi' \models_{TR} \phi$. Furthermore we know by Lemma5 that every model M of P in $\mathcal{TR}$ is also a model M of P in $\mathcal{ETR}$. Moreover since $\models_c$ and $\models_{TR}$ coincide $M, \pi' \models_{TR} \phi$ iff $M, \pi \models_c \phi$ (for π obtained from π' adding an external state E constant). Finally, since π' does not contain external actions in the transitions, then $M, \pi \models_c \phi$ iff $M, \pi \models_{ETR} \phi$. Thus by definition of executional entailment we have that $P, \pi \models_{ETR} \phi$.
$\Leftarrow$:
We want to prove that if $P, \pi \models_{ETR} \phi$ then $P, \pi' \models_{TR} \phi$. Since $P, \pi \models_{ETR} \phi$ we know that for every model M of P (in $\mathcal{ETR}$) then $M, \pi \models_{ETR} \phi$. Moreover since π does not contain external actions in the transitions, then $M, \pi \models_c \phi$ iff $M, \pi \models_{ETR} \phi$. Then since $\models_c$ and $\models_{TR}$ coincide $M, \pi \models_c \phi$ iff $M, \pi' \models_{TR} \phi$. Furthermore we know by Lemma5 that every model M of P in $\mathcal{ETR}$ is also a model M of P in $\mathcal{TR}$ for every interpretation M that is valid in $\mathcal{ETR}$. Then it remains to show that $M', \pi' \models_{TR} \phi$ for every model M' of $\mathcal{TR}$ such that M' is not a valid interpretation in $\mathcal{ETR}$. However it is easy to see by definition of the interpretations of each logic, that this M' can only be non valid interpretation in $\mathcal{ETR}$ and valid in $\mathcal{TR}$ if it more formulas can be true in those paths (e.g. if a is an action in the oracle and M' makes a true in a path that the oracle does not define it so). Then, $\forall M'$ of this form it exists a M'' s.t. $M'' \subset M'$ and M'' is a valid interpretation in $\mathcal{ETR}$ and a model of P. Let's assume M'' as the interpretation as M' that is only different from M' by assuming that actions defined in the oracles are only true whenever they are defined by the oracles as such. Then M'' is not a model of P if it exists a path π_1 s.t. $M'', \pi_1 \models$ *body* but $M'', \pi_1 \not\models$ *head* for some rule $head \leftarrow body$ in P. However by definition of P, $head$ must be an atom defined in $\mathcal{L}_P$ and thus if M' is a model, either M'' $M'', \pi_1 \models head$ or $M'', \pi_1 \not\models body$. By this we can conclude that for every model M in $\mathcal{TR}$ it is true that $M, \pi' \models_{TR} \phi$ and thus by definition of the executional entailment we can conclude that $P, \pi' \models_{TR} \phi$. $\square$

Proof of Theorem 4. By Lemmas 23,22 and 24 $\square$