Transaction Logic with (Complex) Events

Ana Sofia Gomes and José Júlio Alferes*
CENTRIA, Departamento de Informática
Faculdade Ciências e Tecnologias
Universidade Nova de Lisboa
2829-516 Caparica, Portugal

submitted 1 January 2003; revised 1 January 2003; accepted 1 January 2003

Abstract

This work deals with the problem of combining reactive features, such as the ability to respond to events and define complex events, with the execution of ACID transactions over general Knowledge Bases (KBs).

With this as goal, we build on Transaction Logic ($\mathcal{TR}$), a logic precisely designed to model and execute (ACID) transactions in KBs defined by arbitrary logic theories. In it, transactions are written in a logic-programming style, by combining primitive update operations over a general KB, with the usual logic programming connectives and some additional connectives e.g. to express sequence of actions. While $\mathcal{TR}$ is a natural choice to deal with transactions, it remains the question whether $\mathcal{TR}$ can be used to express complex events, but also to deal simultaneously with the detection of complex events and the execution of transactions. In this paper we show that the former is possible while the latter is not. For that, we start by illustrating how $\mathcal{TR}$ can express complex events, and in particular, how SNOOP event expressions can be translated in the logic. Afterwards, we show why $\mathcal{TR}$ fails to deal with the two issues together, and propose Transaction Logic with Events to solve the intended problem. The achieved solution is a non-monotonic conservative extension of $\mathcal{TR}$, which guarantees that every complex event detected in a transaction is necessarily responded. Along with its syntax, model theory and executional semantics, we prove some properties, including that it is indeed a conservative extension, and that it enjoys from important properties of non-monotonic logics, like support.

1 Introduction

Reactivity stands for the ability to detect complex changes (also denoted as events) in the environment and react automatically to them according to some pre-defined rules. This is a pre-requisite of many real-world applications, such as, web-services providing different services depending on external information, multi-agent systems adapting their knowledge and actions according to the happening of changes in the environment, or monitoring systems reacting to information detected by their sensors and issuing actions automatically in response to it. In reactive systems, e.g. in those based on Event-Condition-Action (ECA) languages (Alferes et al. 2011; Bry et al. 2006; Chomicki et al. 2003), the reaction triggered by the detection of a complex event may itself be a complex action, formed e.g. by the sequencial execution of several basic actions. Moreover, we sustain that sometimes reactive systems are also required to execute *transactions* in response to events. For example, consider an airline web-service scenario where an external event arrives stating that a partner airline is on strike for a given time period. Then, the airline must address this event by e.g. rescheduling flights with alternative partners or refund tickets for passengers who do not accept the changes. Clearly, some transactional properties regarding these changes

* The authors thank Michael Kifer for the valuable discussions in a preliminary version of this work.

must be ensured: viz. it can never be the case that a passenger is simultaneously not refunded nor have an alternative flight; or that she is completely refunded and has a rescheduled flight.

Although the possibility of executing transactions, obeying the traditional ACID properties, is of crucial importance in the majority of today's systems, and a must e.g. in database systems, most reactive languages do not deal with it. Some exceptions exist, but are either completely procedural and thus lack from a clear declarative semantics (as e.g. in (Papamarkos et al. 2006)), or have a strong limitation on the expressivity of either the actions or events (as e.g. in (Zaniolo 1995; Lausen et al. 1998)). This is further discussed in Section 4.

In this paper we propose Transaction Logic with Events, $\mathcal{TR}^{ev}$, an extension of $\mathcal{TR}$ (Bonner and Kifer 1993) integrating the ability to reason and execute transactions over very general forms of KBs, with the ability to detect complex events. For this, after a brief overview of $\mathcal{TR}$, we show how it can be used to express and reason about complex events, and in particular, how it can express most SNOOP event operators (Adaikkalavan and Chakravarthy 2006) (Section 2). We proceed by showing why $\mathcal{TR}$ alone is not able to deal with both the detection of complex events and the execution of transactions, and in particular, why it does not guarantee that all complex events detected during the execution of a transaction are responded within that execution. For solving this problem, we define $\mathcal{TR}^{ev}$, its language and model theory (Section 3.1), as well as its executional semantics (Section 3.3). We end with a discussion and related work. This is the extended version of the paper submitted for ICLP'14 containing the proofs of the enunciated results.

2 Using $\mathcal{TR}$ to express complex events

In this section we briefly recall $\mathcal{TR}$'s syntax and semantics with minor syntactic changes from the original, to help distinguish between actions and event occurrences, something that is useful ahead in the paper when extending $\mathcal{TR}$ to deal with reactive transactions and complex events.

Atoms in $\mathcal{TR}$ have the form $p(t_1, \ldots, t_n)$ where p is a predicate symbol and t_i's are terms (variables, constants, function terms). For simplicity, and without loss of generality (Bonner and Kifer 1998), we work with a Herbrand instantiation of the language where the Herbrand base $\mathcal{B}$ is the set of all ground atoms that can be constructed with the functions and constants of the language, and a Herbrand structure is any subset of $\mathcal{B}$. To build complex formulas, $\mathcal{TR}$ uses the classical connectives $\wedge, \vee, \neg, \leftarrow$ and the connectives $\otimes, \Diamond$ denoting serial conjunction and hypothetical execution. Informally, the formula $\phi \otimes \psi$ is an action composed of an execution of ϕ followed by an execution of ψ; and $\Diamond\phi$ tests if ϕ can be executed without materializing the changes. Then, $\phi \wedge \psi$ is the simultaneous execution of ϕ and ψ; while $\phi \vee \psi$ defines the non-deterministic choice of either executing ϕ, ψ or both simultaneously. $\phi \leftarrow \psi$ is a *rule* saying that one way to execute of ϕ is by executing ψ. As in classical logic, $\wedge$ and $\leftarrow$ can be written using $\vee$ and $\neg$, as $\phi \wedge \psi \equiv \neg(\neg\phi \vee \neg\psi)$ and $\phi \leftarrow \psi \equiv \phi \vee \neg\psi$. Finally, we also use the connective ; as it is useful to express common complex events. $\phi; \psi$ says that ψ is true after ϕ but possibly interleaved with other occurrences, and it can be written in $\mathcal{TR}$ syntax as: $\phi \otimes \mathtt{path} \otimes \psi$ where $\mathtt{path} \equiv (\varphi \vee \neg\varphi)$ is a tautology that holds in paths of arbitrary size (Bonner and Kifer 1998).

A very powerful feature of $\mathcal{TR}$ is the incorporation of a pair of oracles $\mathcal{O}^d$ (data oracle) and $\mathcal{O}^t$ (transition oracle) as a parameter of the theory. These define the elementary KB primitives, making possible the separation between the theory of states and updates, with the logic that combines them in transactions. As a result of this separation, $\mathcal{TR}$ does not commit to any particular theory of elementary updates, and can be instantiated using a wide variety of semantics, as e.g.

relational databases, well-founded semantics, action languages, etc. (Bonner and Kifer 1993). These oracles are mappings that assume a set of *state identifiers*. $\mathcal{O}^d$ is a mapping from state identifiers to a set of formulas that hold in that state, and $\mathcal{O}^t$ is a mapping from pairs of state identifiers to sets of formulas that hold in the transition of those states.

Example 1 (*Relational Oracle - (Bonner and Kifer 1993)*)
A KB made of a relational database can be modeled by having states represented as sets of ground atomic formulas. The data oracle simply returns all these formulas, i.e., $\mathcal{O}^d(D) = D$. Moreover, for each predicate p in the KB, the transition oracle defines $p.ins$ and $p.del$, representing the insertion and deletion of p, respectively. Formally, $p.ins \in \mathcal{O}^t(D_1, D_2)$ iff $D_2 = D_1 \cup \{p\}$ and, $p.del \in \mathcal{O}^t(D_1, D_2)$ iff $D_2 = D_1 \backslash \{p\}$. SQL-style bulk updates can also be defined by $\mathcal{O}^t$.

Example 2 (*Moving objects - $\mathcal{TR}$*)
As an illustration of $\mathcal{TR}$, assume the previous oracle definition and the action $move(O, X, Y)$ defining the relocation of object O from position X into position Y. In such a KB, states are defined using the predicates $location(O, P)$ saying that object O is in position P, and $clear(X)$ stating that X is clear to receive an object. In $\mathcal{TR}$, the move (trans)action can be expressed by:

$$move(O, X, Y) \leftarrow location(O, X) \otimes clear(Y) \otimes localUpdt(O, X, Y)$$
$$localUpdt(O, X, Y) \leftarrow location(O, X).del \otimes location(O, Y).ins \otimes clear(Y).del \otimes clear(X).ins$$

$\mathcal{TR}$'s theory is built upon the notion of sequences of states denoted as *paths*. Formulas are evaluated over paths, and truth in $\mathcal{TR}$ means *execution*: a formula is said to succeed over a path, if that path represents a valid execution for that formula. Although not part of the original $\mathcal{TR}$, here paths' state transitions are labeled with information about what (atomic occurrences) happen in the transition of states. Precisely, paths have the form $\langle D_0,^{O_1} D_2,^{O_2} \ldots,^{O_k} D_k\rangle$, where D_i's are states and O_i's are labels (used later to annotate atomic event occurrences).

As usual, satisfaction of complex formulas is based on interpretations. These define what atoms are true in what paths, by mapping every path to a Herbrand structure. However, only the mappings compliant with the specified oracles are considered as interpretations:

Definition 1 (*Interpretation*)
An interpretation is a mapping M assigning a classical Herbrand structure (or $\top$[1]) to every path, with the following restrictions (where D_is are states, and φ a formula):

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

In point 2 we additionally (i.e., when compared to the original definition) force $\mathbf{o}(\varphi)$ to belong to the same path where the primitive action φ is made true by the oracle, something that later (in Section 3) will help detect events associated with primitive actions, like "on insert/delete".

Next, we define operations on paths, and satisfaction of complex formulas over general paths.

Definition 2 (*Path Splits, Subpaths and Prefixes*)
Let π be a k-path, i.e. a path of length k of the form $\langle D_1,^{O_1} \ldots,^{O_{k-1}} D_k\rangle$. A *split* of π is any pair of subpaths, π_1 and π_2, such that $\pi_1 = \langle D_1,^{O_1} \ldots,^{O_{i-1}} D_i\rangle$ and $\pi_2 = \langle D_i,^{O_i} \ldots,^{O_{k-1}} D_k\rangle$ for some i $(1 \leq i \leq k)$. In this case, we write $\pi = \pi_1 \circ \pi_2$.
A subpath π' of π is any subset of states and annotations of π where both the order of the states and their annotations is preserved. A prefix π_1 of π is any subpath of π sharing the initial state.

[1] For not having to consider partial mappings, besides formulas, interpretations can also return the special symbol $\top$. The interested reader is referred to (Bonner and Kifer 1993) for details.

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

1. **Base Case:** $M, \pi \models_{\mathcal{TR}} \phi$ iff $\phi \in M(\pi)$ for every event occurrence ϕ
2. **Negation:** $M, \pi \models_{\mathcal{TR}} \neg\phi$ iff it is not the case that $M, \pi \models_{\mathcal{TR}} \phi$
3. **Disjunction:** $M, \pi \models_{\mathcal{TR}} \phi \vee \psi$ iff $M, \pi \models_{\mathcal{TR}} \phi$ or $M, \pi \models_{\mathcal{TR}} \psi$.
4. **Serial Conjunction:** $M, \pi \models_{\mathcal{TR}} \phi \otimes \psi$ iff there exists a split $\pi_1 \circ \pi_2$ of π s.t. $M, \pi_1 \models_{\mathcal{TR}} \phi$ and $M, \pi_2 \models_{\mathcal{TR}} \psi$
5. **Executional Possibility:** $M, \pi \models_{\mathcal{TR}} \Diamond\phi$ iff π is a 1-path of the form $\langle D \rangle$ for some state D and $M, \pi' \models_{\mathcal{TR}} \phi$ for some path π' that begins at D.

Models and logical entailment are defined as usual. An interpretation models/satisfies a set of rules if each rule is satisfied in every possible path, and an interpretation models a rule in a path, if whenever it satisfies the antecedent, it also satisfies the consequent.

Definition 4 (*Models, and Logical Entailment*)
An interpretation M is a *model* of a formula ϕ iff for every path π, $M, \pi \models_{\mathcal{TR}} \phi$. M is a model of a set of rules P (denoted $M \models_{\mathcal{TR}} P$) iff it is a model of every rule in P.
ϕ is said to logical entail another formula ψ iff every model of ϕ is also a model of ψ.

Logical entailment is useful to define general equivalence and implication of formulas that express properties like "transaction ϕ is equivalent to transaction ψ" or "whenever transaction ψ is executed, ψ' is also executed". Moreover, if instead of transactions, we view the propositions as representing event occurrences, this entailment can be used to express complex events. For instance, imagine we want to state a complex event $alarm$ triggered, e.g. whenever event ev_1 occurs after the events ev_2 and ev_3 simultaneously occur. This can be expressed in $\mathcal{TR}$ as:

$$\mathbf{o}(alarm) \leftarrow (\mathbf{o}(e_2) \wedge \mathbf{o}(e_3)); \mathbf{o}(e_1) \tag{1}$$

In every model of this formula, whenever there is a (sub)path where both $\mathbf{o}(e_2)$ and $\mathbf{o}(e_3)$ are true, followed by a (sub)path where $\mathbf{o}(e_1)$ holds, then $\mathbf{o}(alarm)$ is true in the *whole* path.

Other complex event definitions are possible, and in fact we can encode most of SNOOP (Adaikkalavan and Chakravarthy 2006) operators in $\mathcal{TR}$. This is shown in Theorem 1 where, for a given history of past event occurrences, we prove that if an event expression is true in SNOOP, then there is a translation into a $\mathcal{TR}$ formula which is also true in that history. Since a SNOOP history is a set of atomic events associated with discrete points in time, the first step is to build a $\mathcal{TR}$ path expressing such history. We construct it as a sequence of state identifiers labeled with time, where time point i takes place in the transition of states $\langle s_i, s_{i+1} \rangle$, and only consider interpretations M over such a path that are *compatible* with SNOOP's history, i.e. such that, for every atomic event that is true in a time i, M makes the same event true in the path $\langle s_i, s_{i+1} \rangle$.

Theorem 1 (*SNOOP Algebra and $\mathcal{TR}$*)
Let E be a *SNOOP* algebra expression without periodic and aperiodic operators, H be a history containing the set of all SNOOP primitive events $e^i_j[t_1]$ that have occurred over the time interval t_1, t_{max}, and $\langle s_1, \ldots s_{max+1} \rangle$ be a path with size $t_{max} - t_1 + 1$. Let τ be the following function:

Primitive:	$\tau(E) = \mathbf{o}(E)$ where E is a primitive event
Sequence:	$\tau(E_1; E_2) = \tau(E_1) \otimes \mathtt{path} \otimes \tau(E_2)$
Or:	$\tau(E_1 \triangledown E_2) = \tau(E_1) \vee \tau(E_2)$
AND:	$\tau(E_1 \triangle E_2) = [(\tau(E_1) \otimes \mathtt{path}) \wedge (\mathtt{path} \otimes \tau(E_2))] \vee [(\tau(E_2) \otimes \mathtt{path}) \wedge (\mathtt{path} \otimes \tau(E_1))]$
NOT:	$\tau(\neg(E_3)[E_1, E_2]) = (\tau(E_1) \otimes \mathtt{path} \otimes \tau(E_2)) \wedge \neg(\mathtt{path} \otimes \tau(E_2) \otimes \mathtt{path})$

Then:

If $[t_i, t_f] \in E[H]$ then $\forall M$ *compatible with* H, $M, \langle s_{t_i}, \ldots, s_{t_{f+1}} \rangle \models_{\mathcal{TR}} \tau(E)$

where, cf. (Adaikkalavan and Chakravarthy 2006), $E[H]$ is the set of time intervals (t_i, t_f) where E occurs over H in an unrestricted context, and where M is compatible with H if, for each $e^i_j[t_i] \in H$: $M, \langle s_{t_i}, s_{t_{i+1}} \rangle \models_{\mathcal{TR}} \mathbf{o}(e_j)$.

Proof
We prove by induction on the structure of E.

Base Primitive: $E = E_j$
Assume $[t_i, t_f] \in E_j[H]$ then since E_j is a primitive event, it must exist a $e^i_j[t_i] \in H$ and $t_i = t_f$. Then, since M is compatible with H we have that $M, \langle s_{t_i}, s_{t_{i+1}} \rangle \models_{\mathcal{TR}} \mathbf{o}(e_j)$ as intended.

Sequence: $E = E_1; E_2$
Assume $[t_i, t_f] \in (E_1; E_2)[H]$ then, by SNOOP's definition we know that $\exists t_{f1}, t_{i2}$ s.t. $t_i \leq t_{f1} < t_{i2} \leq t_f$ and $[t_i, t_{f1}] \in E_1[H]$, $[t_{i2}, t_f] \in E_2[H]$. Then, we can apply the I.H. and conclude that ∀M compatible with H, $M, \langle s_{t_i}, \ldots, s_{t_{f1+1}} \rangle \models_{\mathcal{TR}} \tau(E_1)$ and that $M, \langle s_{t_{i2}}, \ldots, s_{t_{f+1}} \rangle \models_{\mathcal{TR}} \tau(E_2)$.
Additionally $\langle s_{t_i}, \ldots, s_{t_{f1+1}} \rangle$ and $\langle s_{t_{i2}}, \ldots, s_{t_f} \rangle$ are subsets of $\langle s_1, \ldots s_{max+1} \rangle$ where the former subpath occurs before the latter. As a result, by the satisfaction relation definition of $\mathcal{TR}$, we know that $M, \langle s_{t_i}, \ldots, s_{t_{f1+1}}, \ldots s_{t_{i2}}, \ldots, s_{t_f} \rangle \models_{\mathcal{TR}} \tau(E_1) \otimes \mathtt{path} \otimes \tau(E_2)$ which is equivalent to $M, \langle s_{t_i}, \ldots, s_{t_{f1+1}}, \ldots s_{t_{i2}}, \ldots, s_{t_f} \rangle \models_{\mathcal{TR}} \tau(E_1); \tau(E_2)$

Or: $E = E_1 \triangledown E_2$
Assume $[t_i, t_f] \in (E_1 \triangledown E_2)[H]$ then, by SNOOP's definition we know that either $[t_i, t_f] \in E_1[H]$ or $[t_i, t_f] \in E_2[H]$. Then by I.H. we can conclude that ∀M compatible with H, $M, \langle s_{t_i}, \ldots, s_{t_{f+1}} \rangle \models_{\mathcal{TR}} \tau(E_1)$ or $M, \langle s_{t_i}, \ldots, s_{t_{f+1}} \rangle \models_{\mathcal{TR}} \tau(E_2)$. Finally, by the satisfaction relation definition of $\mathcal{TR}$ we know that $M, \langle s_{t_i}, \ldots, s_{t_{f+1}} \rangle \models_{\mathcal{TR}} \tau(E_1) \vee \tau(E_2)$.

AND: $E = \tau(E_1 \triangle E_2)$
Assume $[t_i, t_f] \in (E_1 \triangledown E_2)[H]$ then, by SNOOP's definition we know that $\exists t_{f'}, t_{i'}$ s.t. $t_i \leq t_{f'} \leq t_{i'} \leq t_f$ and either one of the two cases is true:

- $[t_i, t_{f'}] \in E_1[H]$, $[t_{i'}, t_f] \in E_2[H]$; or
- $[t_i, t_{f'}] \in E_2[H]$, $[t_{i'}, t_f] \in E_1[H]$

Let's assume (1). Then by I.H. we know that ∀M compatible with H, $M, \langle s_{t_i}, \ldots, s_{t_{f'}} \rangle \models_{\mathcal{TR}} \tau(E_1)$ and $M, \langle s_{t_{i'}}, \ldots, s_{t_f} \rangle \models_{\mathcal{TR}} \tau(E_2)$. Since $t_{f'} \leq t_{i'}$ we have $M, \langle s_{t_i}, \ldots, s_{t_f} \rangle \models_{\mathcal{TR}} (\tau(E_1) \otimes \mathtt{path})$ and $M, \langle s_{t_i}, \ldots, s_{t_f} \rangle \models_{\mathcal{TR}} (\mathtt{path} \otimes \tau(E_2))$. Thus $M, \langle s_{t_i}, \ldots, s_{t_f} \rangle \models_{\mathcal{TR}} (\tau(E_1) \otimes \mathtt{path}) \wedge (\mathtt{path} \otimes \tau(E_2))$ and $M, \langle s_{t_i}, \ldots, s_{t_f} \rangle \models_{\mathcal{TR}} [(\tau(E_1) \otimes \mathtt{path}) \wedge (\mathtt{path} \otimes \tau(E_2))] \vee [(\tau(E_2) \otimes \mathtt{path}) \wedge (\mathtt{path} \otimes \tau(E_1))]$
Let's assume (2). Then by I.H. we know that ∀M compatible with H, $M, \langle s_{t_i}, \ldots, s_{t_{f'}} \rangle \models_{\mathcal{TR}} \tau(E_2)$ and $M, \langle s_{t_{i'}}, \ldots, s_{t_f} \rangle \models_{\mathcal{TR}} \tau(E_1)$. Since $t_{f'} \leq t_{i'}$ we have $M, \langle s_{t_i}, \ldots, s_{t_f} \rangle \models_{\mathcal{TR}} (\tau(E_2) \otimes \mathtt{path})$ and $M, \langle s_{t_i}, \ldots, s_{t_f} \rangle \models_{\mathcal{TR}} (\mathtt{path} \otimes \tau(E_1))$. Thus $M, \langle s_{t_i}, \ldots, s_{t_f} \rangle \models_{\mathcal{TR}} (\tau(E_2) \otimes \mathtt{path}) \wedge (\mathtt{path} \otimes \tau(E_1))$ and $M, \langle s_{t_i}, \ldots, s_{t_f} \rangle \models_{\mathcal{TR}} [(\tau(E_1) \otimes \mathtt{path}) \wedge (\mathtt{path} \otimes \tau(E_2))] \vee [(\tau(E_2) \otimes \mathtt{path}) \wedge (\mathtt{path} \otimes \tau(E_1))]$.

NOT: $E = \neg(E_3)[E_1, E_2]$
Assume $[t_i, t_f] \in \neg(E_3)[E_1, E_2][H]$ then, by SNOOP's definition we know that $\exists t_{f1}, t_{i2}$ s.t. $t_i \leq t_{f1} < t_{i2} \leq t_f$, $[t_i, t_{f1}] \in E_1[H]$, $[t_{i2}, t_f] \in E_2[H]$ and it is not the case that $\exists t_{i3}, t_{f3}$ where $t_i \leq t_{i3} \leq t_{f3} \leq t_f$ and $[t_{i3}, t_{f3}] \in E(H)$ From the case proof of Sequence, we know that the fist part entails that $M, \langle s_{t_i}, \ldots, s_{t_{f1+1}}, \ldots s_{t_{i2}}, \ldots, s_{t_f} \rangle \models_{\mathcal{TR}} \tau(E_1); \tau(E_2)$.

Additionally let's assume that $\exists t_{i3}, t_{f3}$ where $t_i \leq t_{i3} \leq t_{f3} \leq t_f$ and $[t_{i3}, t_{f3}] \in E(H)$. Then by I.H. we know that $M, \langle s_{t_{i3}}, \ldots, s_{t_{f3+1}} \rangle \models_{\mathcal{TR}} \tau(E_3)$. Since $t_i \leq t_{i3} \leq t_{f3} \leq t_f$ then $M, \langle s_{t_i}, \ldots, s_{t_{f+1}} \rangle \models_{\mathcal{TR}} \texttt{path} \otimes \tau(E_3) \otimes \texttt{path}$.
Additionally, if it is not the case that $M, \langle s_{t_i}, \ldots, s_{t_{f+1}} \rangle \models_{\mathcal{TR}} \texttt{path} \otimes \tau(E_3) \otimes \texttt{path}$, then we can conclude $M, \langle s_{t_i}, \ldots s_{t_{f+1}} \rangle \models_{\mathcal{TR}} \neg(\texttt{path} \otimes \tau(E_3) \otimes)$. Based on this we know that $M, \langle s_{t_i}, \ldots, s_{t_{f+1}} \rangle \models_{\mathcal{TR}} (\tau(E_1); \tau(E_2)) \wedge (\texttt{path} \otimes \tau(E_3) \otimes \texttt{path}$

□

Besides the logical entailment, $\mathcal{TR}$ also provides the notion of executional entailment for reasoning about properties of a *specific* execution path.

Definition 5 (*Executional Entailment*)
Let P be a set of rules, ϕ a formula, and $D_0,^{O_1} \ldots,^{O_n} D_n$ a path.
$P, (D_0,^{O_1} \ldots,^{O_n} D_n) \models \phi$ $(\star)$ iff for every model M of P, $M, \langle D_0,^{O_1} \ldots,^{O_n} D_n \rangle \models \phi$.
Additionally, $P, D_0 - \models \phi$ holds, if there is a path $D_0,^{O_1} \ldots,^{O_n} D_n$ that makes $(\star)$ true.

$P, (D_0,^{O_1} \ldots,^{O_n} D_n) \models \phi$ says that a successful execution of transaction ϕ respecting the rules in P, can change the KB from state D_0 into D_n with a sequence of occurrences $O_1, \ldots, O_n$. E.g., in the Example 2 (with obvious abbreviations), the statement $P, (\{cl(t), l(c,o)\},^{\mathbf{o}(l(c,o).del)} \{cl(t)\},^{\mathbf{o}(l(c,t).ins)} \{cl(t), l(c,t)\},^{\mathbf{o}(cl(t).del)} \{l(c,t)\},^{\mathbf{o}(cl(o).ins)} \{l(c,t), cl(o)\}) \models move(c,o,t)$ means that a possible result of executing the transaction $move(c, oven, table)$ starting in the state $\{clear(table), loc(c, oven)\}$ is the path with those 5 states, ending in $\{loc(c, table), clear(oven)\}$.

This entailment has a corresponding proof theory (Bonner and Kifer 1993) which, for a subset of $\mathcal{TR}$, is capable of *constructing* such a path given a program, a $\mathcal{TR}$ formula, and an initial state. I.e. a path where the formula can be executed respecting the ACID properties. If no such path exists, then the transaction fails, and nothing is built after the initial state.

3 $\mathcal{TR}^{ev}$: combining the execution of transactions with complex event detection

Reactive languages need to express behaviors like: "on alarm do action a_1 followed by action a_2", where the actions $a_1 \otimes a_2$ may define a transaction, and alarm is e.g. the complex event in (1). Clearly, $\mathcal{TR}$ can individually express and reason about transaction $a_1 \otimes a_2$, and its complex event. So, the question is whether it can deal with both simultaneously. For that, two important issues must be tackled: 1) how to model the triggering behavior of reactive systems, where the occurrence of an event drives the execution of a transaction in its response; 2) how to model the transaction behavior that prevents transactions to commit until all occurring events are responded.

Regarding 1), (Bonner et al. 1993) shows that simple events can be triggered in $\mathcal{TR}$ as:

$$\begin{array}{l} p \leftarrow body \otimes ev \\ ev \leftarrow \mathbf{r}(ev) \end{array} \tag{2}$$

With such rules, in all paths that make p true (i.e. in all executions of transaction p) the event ev is triggered/fired (after the execution of some arbitrary $body$), and ev's response, $\mathbf{r}(ev)$, is executed. Note that, both $\mathbf{r}(e)$ and $body$ can be defined as arbitrary formulas.

But, this is just a very simple and specific type of event: atomic events that are explicitly triggered by a transaction defined in the program. In general, atomic events can also arrive as external events, or because some primitive action is executed in a path (e.g. as the database triggers - "on insert/on delete"). Triggering external events in $\mathcal{TR}$ is easy, and can be done by considering the paths that make the external event true. E.g., if one wants to respond to an external event ev from an initial state, all we need to do is find the paths π starting in that state, s.t. $P, \pi \models ev$, where P includes the last rule above plus the rules stating how to respond to ev.

The occurrences of primitive actions can be tackled by Point 2 of Def. 1, and the occurrence of complex events can be defined as prescribed in Section 2. However, the above approach of (Bonner et al. 1993) does not help for driving the execution of an event response when such occurrences become true. For instance, the ECA-rule before could be stated as:

$$\begin{array}{l}\mathbf{o}(alarm) \leftarrow (\mathbf{o}(e_2) \wedge \mathbf{o}(e_3)); \mathbf{o}(e_1)\\ \mathbf{r}(alarm) \leftarrow a_1 \otimes a_2\end{array}$$

But this does not drive the execution of $\mathbf{r}(alarm)$ when $\mathbf{o}(alarm)$ holds; one has further to force that whenever $\mathbf{o}(alarm)$ holds, $\mathbf{r}(alarm)$ must be made true subsequently. Of course, adding a rule $\mathbf{r}(alarm) \leftarrow \mathbf{o}(alarm)$ would not work: such rule would only state that, one alternative way to satisfy the response of alarm is to make its occurrence true. And for that, it would be enough to satisfy $\mathbf{o}(alarm)$ to make $\mathbf{r}(alarm)$ true, which is not what is intended.

Clearly, this combination implies two different types of formulas with two very different behaviors: the *detection* of events which are tested for occurrence w.r.t. a past history; and the *execution* of transactions as a response to them, which intends to construct paths where formulas can succeed respecting transactional properties. This has to be reflected in the semantics and these formulas should be evaluated differently accordingly to their nature.

Regarding 2), as in database triggers, transaction's execution must depend on the events triggered. Viz., an event occurring during a transaction execution can delay that transaction to commit/succeed until the event response is successfully executed, and the failure of such response should imply the failure of the whole transaction. Encoding this behavior requires that, if an event occurs during a transaction, then its execution needs to be *expanded* with the event response. Additionally, this also precludes transactions to succeed in paths where an event occurs and is not responded (even if the transaction would succeed in that path if the event did not existed).

For addressing theses issues, below we define $\mathcal{TR}^{ev}$. It evaluates event formulas and transaction formulas differently, using two distinct relations (respectively $\models_{\mathcal{TR}}$ and $\models$), and occurrences and responses are syntactic represented w.r.t. a given event name e, as $\mathbf{o}(e)$ and $\mathbf{r}(e)$, respectively. $\models$ requires transactions to be satisfied in expanded paths, where every occurring event (i.e. made true by $\models_{\mathcal{TR}}$) is properly responded. We shall see that $\models$ implies a non-monotonic relation, as truth of transactions on a path may change if a new event occurrences is learned to be true.

3.1 $\mathcal{TR}^{ev}$ Syntax and Model Theory

$\mathcal{TR}^{ev}$ alphabet contains an infinite number of constants $\mathcal{C}$, function symbols $\mathcal{F}$, variables $\mathcal{V}$ and predicate symbols $\mathcal{P}$. To make it possible to evaluate events and transactions differently, predicate symbols are further partitioned into transaction names ($\mathcal{P}_t$), event names ($\mathcal{P}_e$), and oracle primitives ($\mathcal{P}_{\mathcal{O}}$) and, as with $\mathcal{TR}$, we work with the Herbrand instantiation of the language.

Formulas in $\mathcal{TR}^{ev}$ are partitioned into transaction formulas and event formulas. *Event formulas* denote formulas meant to be *detected* and are either an event occurrence, or an expression defined inductively as $\neg\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, $\phi \otimes \psi$, or $\phi; \psi$ where ϕ and ψ are event formulas. An *event occurrence* is of the form $\mathbf{o}(\varphi)$ s.t. $\varphi \in \mathcal{P}_e$ or $\varphi \in \mathcal{P}_{\mathcal{O}}$, and where the latter corresponds to the case where occurrences arise from detecting the execution of an oracle defined primitive. Note that, we preclude the usage of the hypothetical execution operator $\Diamond$ in event formulas, as it would make little sense to detect occurrences based on what could possibly be executed.

Transaction formulas are formulas that can be *executed*, and are either a transaction atom, or an expression defined inductively as $\neg\phi$, $\Diamond\phi$, $\phi \wedge \psi$, $\phi \vee \psi$, or $\phi \otimes \psi$. A *transaction atom* is either a proposition in $\mathcal{P}_t$, $\mathcal{P}_{\mathcal{O}}$, $\mathbf{r}(\varphi)$ where $\varphi \in \mathcal{P}_{\mathcal{O}} \cup \mathcal{P}_e$, or $\mathcal{P}_e$. I.e. a transaction atom is either a transaction name, an oracle defined primitive, the response of an event, or an event name. The latter corresponds to the (trans)action of *explicitly* triggering an event directly in a transaction as in (2) or as an external event. As we shall see (Def. 2) explicitly triggering an event changes the path of execution (by asserting the information that the event has happened in the current state) and, as such, is different from simply inferring (or detecting) what events hold given a past path.

Finally, rules have the form $\varphi \leftarrow \psi$ and can be transaction or (complex) event rules. In a transaction rule φ is a transaction atom and ψ a transaction formula; whereas in an event rule φ is an event occurrence and ψ is a event formula. A *program* is a set of transaction and event rules.

Importantly, besides the data and transition oracles, $\mathcal{TR}^{ev}$ is also parametric to a $choice$ function defining what event should be selected at a given time in case of conflict. Since defining what event should be picked from the set of occurring events depends on the application in mind, to be useful for a a wide spectrum of applications, $\mathcal{TR}^{ev}$ does not commit to any particular definition, encapsulating it in function $choice$. This will be further elaborated in Section 3.2.

As a reactive system, $\mathcal{TR}^{ev}$ receives a series of external events which may cause the execution of transactions in response. This is defined as $P, D_0 - \models e_1 \otimes \ldots \otimes e_k$, where D_0 is the initial KB state and $e_1 \otimes \ldots \otimes e_k$ is the sequence of external events that arrive to the system. Here, we want to find the path $D_0, ^{O_1} \ldots, ^{O_n} D_n$ encoding a KB evolution that responds to $e_1 \otimes \ldots \otimes e_k$.

As mentioned, triggering explicit events is a transaction formula encoding the *action* of making an occurrence explicitly true. This is handled by the definition of interpretation, in a similar way to how atomic events defined by oracles primitives are made true:

Definition 6 (*$\mathcal{TR}^{ev}$ interpretations*)
A $\mathcal{TR}^{ev}$ interpretation is a $\mathcal{TR}$ interpretation that additionally satisfies the restriction:

3. $\mathbf{o}(e) \in M(\langle D, ^{\mathbf{o}(e)} D \rangle)$ if $e \in \mathcal{P}_e$

We can now define the satisfaction of complex formulas, and then models of a program. As mentioned, event formulas are evaluated w.r.t. the relation $\models_{\mathcal{TR}}$ specified in Def. 3. Transaction formulas are evaluated w.r.t. the relation $\models$ which requires formulas to be true in *expanded paths*, in which every occurring event is responded (something dealt by $\mathtt{exp}_M(\pi)$, defined below).

Definition 7 (*Satisfaction of Transaction Formulas*)
Let M be an interpretation, π a path, ϕ transaction formula. If $M(\pi) = \top$ then $M, \pi \models \phi$; else:

1. **Base Case:** $M, \pi \models p$ iff $\exists \pi'$ prefix of π s.t. $p \in M(\pi')$ and $\pi = \mathtt{exp}_M(\pi')$, for every transaction atom p where $p \notin \mathcal{P}_e$.
2. **Event Case:** $M, \pi \models e$ iff $e \in \mathcal{P}_e$, $\exists \pi'$ prefix of π s.t. $M, \pi' \models_{\mathcal{TR}} \mathbf{o}(e)$ and $\pi = \mathtt{exp}_M(\pi')$.
3. **Negation:** $M, \pi \models \neg\phi$ iff it is not the case that $M, \pi \models \phi$
4. **Disjunction:** $M, \pi \models \phi \vee \psi$ iff $M, \pi \models \phi$ or $M, \pi \models \psi$.
5. **Serial Conjunction:** $M, \pi \models \phi \otimes \psi$ iff $\exists \pi'$ prefix of π and some split $\pi_1 \circ \pi_2$ of π' such that $M, \pi_1 \models \phi$ and $M, \pi_2 \models \psi$ and $\pi = \mathtt{exp}_M(\pi')$.
6. **Executional Possibility:** $M, \pi \models \Diamond\phi$ iff π is a 1-path of the form $\langle D \rangle$ for some state D and $M, \pi' \models \phi$ for some path π' that begins at D.

Definition 8 (*Model of a Program*)
An interpretation M is a *model* of a transaction formula (resp. event formula) ϕ iff for every path π, $M, \pi \models \phi$ (resp. $M, \pi \models_{\mathcal{TR}} \phi$). M is a model of a program P (denoted $M \models P$) iff it is a model of every (transaction and complex event) rule in P.

We have yet to defined $\mathtt{exp}_M(\pi)$, a function that, given a path with possibly unanswered events, expands it with the result of responding to those events. The definition of this function must perforce have some procedural nature: it must start by detecting which are the unanswered events; pick one of them, according to a given *choice* function; then expand the path with the response of the chosen event. The response to this event may, in turn, generate the occurrence of further events. So, this process must be iterated until no more unanswered events exist.

The response operator $\mathcal{R}_M$, defined below, computes one step of this iteration. At each iteration, $\mathcal{R}_M$ receives a path π and returns a path π'. If π has unanswered events w.r.t. M, then π' is an extension of π where one of the events unanswered in π is now responded in π'; otherwise, $\pi' = \pi$. As stated, since complex events exist, nothing prevents $\mathcal{R}_M(\pi)$ to return a path π' with more unanswered events than π. Moreover, it may not be possible to address all events in a finite path, and thus, $\mathcal{R}_M$ may not have a fixed-point. In fact, non-termination is a known problem of reactive systems, and is often undecidable for the general case (Bailey et al. 2004). However, if termination is possible, then a fixed-point exists and each iteration of $\mathcal{R}_M$ is an approximation of the expansion operator $\mathtt{exp}_M$.

Definition 9 (*Expansion of a Path*)
For a path π_1 and an interpretation M, the response operator $\mathcal{R}_M(\pi_1)$ is defined as follows:

$$\mathcal{R}_M(\pi_1) = \begin{cases} \pi_1 \circ \pi_2 & \textbf{if } \textit{choice}(M, \pi_1) = e \text{ and } M, \pi_2 \models \mathbf{r}(e) \\ \pi_1 & \textbf{if } \textit{choice}(M, \pi_1) = \epsilon \end{cases}$$

The expansion of a path π is $\mathtt{exp}_M(\pi) = \uparrow \mathcal{R}_M(\pi)$.

Before discussing the *choice* function, we first formalize the effect of $\mathtt{exp}_M(\pi)$ on satisfying transaction formulas, and exemplify the semantics in examples where the *choice* is not crucial (viz. there is always at most one event to choose).

Lemma 1
Let M be an interpretation, ϕ be a transaction formula without negation, and π be a path.

$$\text{If } M, \pi \models \phi \text{ then } \mathtt{exp}_M(\pi) = \pi$$

Proof
Immediately from Definition 2 □

Example 3

$$\begin{array}{ll} p \leftarrow a.ins & \\ \mathbf{r}(e_1) \leftarrow c.ins & (P_3) \end{array} \qquad \begin{array}{ll} p \leftarrow a.ins & \\ \mathbf{r}(e_1) \leftarrow c.ins & (P_4) \\ \mathbf{o}(e_1) \leftarrow \mathbf{o}(a.ins) & \end{array}$$

Consider the programs[2] P_3 and P_4. In P_3, p holds in the path $\langle \{\}, ^{\mathbf{o}(a.ins)} \{a\} \rangle$. This is true since all interpretations must comply with the oracles and thus $\forall M$: $a.ins \in M(\langle \{\}, ^{\mathbf{o}(a.ins)} \{a\} \rangle)$ implying $M, \langle \{\}, ^{\mathbf{o}(a.ins)} \{a\} \rangle \models a.ins$. Assuming that M is a model of P_3, then it satisfies the rule $p \leftarrow a.ins$, which means that $p \in M(\langle \{\}, ^{\mathbf{o}(a.ins)} \{a\} \rangle)$ and $M, \langle \{\}, ^{\mathbf{o}(a.ins)} \{a\} \rangle \models p$.

However, since $\mathbf{o}(e_1) \leftarrow \mathbf{o}(a.ins) \in P_4$ and $\forall M. \mathbf{o}(a.ins) \in M(\langle \{\}, ^{\mathbf{o}(a.ins)} \{a\} \rangle)$, for M to be a model of P_4, then $\mathbf{o}(e_1) \in M(\langle \{\}, ^{\mathbf{o}(a.ins)} \{a\} \rangle)$. Since e_1 has a response defined, then in path $\langle \{\}, ^{\mathbf{o}(a.ins)} \{a\} \rangle$ the occurrence e_1 is unanswered and both the transactions p and $a.ins$

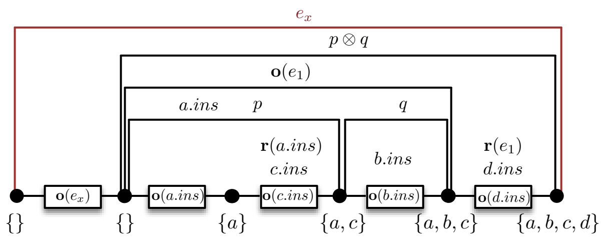
[2] For brevity, in this and the following examples we assume the rule $\mathbf{r}(p) \leftarrow \mathtt{true}$ to appear in every program for every primitive action p defined in the signature of the oracles, unless when stated otherwise. I.e., we assume the responses of events inferred from primitive actions to hold trivially whenever their rules do not appear explicitly in the program.

cannot succeed in that path. Namely, $\mathbf{o}(e_1)$ constrains the execution of *every* transaction in the path $\langle \{\},^{\mathbf{o}(a.ins)} \{a\}\rangle$ and, for transaction formulas to succeed, such path needs to be *expanded* with e_1's response. Since, $\mathtt{exp}_M(\langle \{\},^{a.ins} \{a\}\rangle) = \langle \{\},^{\mathbf{o}(a.ins)} \{a\},^{\mathbf{o}(c.ins)} \{a,c\}\rangle$ then, both transactions p and $a.ins$ succeed in the *longer* path $\langle \{\},^{\mathbf{o}(a.ins)} \{a\},^{\mathbf{o}(c.ins)} \{a,c\}\rangle$, i.e. for an M model of P_4: $M, \langle \{\},^{\mathbf{o}(a.ins)} \{a\},^{\mathbf{o}(c.ins)} \{a,c\}\rangle \models p$ and $M, \langle \{\},^{\mathbf{o}(a.ins)} \{a\},^{\mathbf{o}(c.ins)} \{a,c\}\rangle \models a.ins$. This illustrates the non-monotonicity of $\mathcal{TR}^{ev}$, viz. that adding a new event definition to P_3 falsifies the transaction formulas p and $a.ins$ in paths where they were previously true.

As in $\mathcal{TR}$ when talking about transactions, in $\mathcal{TR}^{ev}$ every formula that is meant to be executed, is meant to be executed as a transaction. As such, the primitive $a.ins$ in example P_4 cannot succeed in the path $\langle \{\},^{\mathbf{o}(a.ins)} \{a\}\rangle$ since there exist unanswered events in that path. However, note that $a.ins$ belongs to every interpretation M of that path (due to the restrictions in Def. 1). Thus the primitive $a.ins$ is true in $\langle \{\},^{\mathbf{o}(a.ins)} \{a\}\rangle$ although the transaction $a.ins$ is not.

Example 4

$p \leftarrow a.ins$
$q \leftarrow b.ins$
$\mathbf{r}(e_x) \leftarrow p \otimes q$
$\mathbf{r}(e_1) \leftarrow d.ins$
$\mathbf{r}(a.ins) \leftarrow c.ins$
$\mathbf{o}(e_1) \leftarrow \mathbf{o}(a.ins) \otimes \mathbf{o}(b.ins)$



The right-hand side figure illustrates a satisfaction of the external event e_x. The occurrence of e_x forces the satisfaction of the transaction $p \otimes q$, which is true if both its "subformulas" (p and q) are satisfied over smaller paths. Note that, by definition of the relation $\models$, all occurrences detected over the independent paths that satisfy p and q are already responded in those paths. Thus, we need only to cater for the events triggered due to the serial conjunction. Here, for a model M of the program, $M, \langle \{\},^{\mathbf{o}(a.ins)} \{a\},^{\mathbf{o}(c.ins)} \{a,c\}\rangle \models p$ and $M, \langle \{a,c\},^{\mathbf{o}(b.ins)} \{a,b,c\}\rangle \models q$. Further, the rule $\mathbf{o}(e_1) \leftarrow \mathbf{o}(a.ins) \otimes \mathbf{o}(b.ins)$ defines one pattern for the occurrence of e_1 which constrains the execution of transaction $p \otimes q$ and forces the expansion of the path to satisfy $\mathbf{r}(e_1)$. Consequently, $M, \langle \{\},^{\mathbf{o}(a.ins)} \{a\},^{\mathbf{o}(c.ins)} \{a,c\},^{\mathbf{o}(b.ins)} \{a,b,c\},^{\mathbf{o}(d.ins)} \{a,b,c,d\}\rangle \models p \otimes q$, and $M, \langle \{\},^{\mathbf{o}(e_x)} \{\},^{\mathbf{o}(a.ins)} \{a\},^{\mathbf{o}(c.ins)} \{a,c\},^{\mathbf{o}(b.ins)} \{a,b,c\},^{\mathbf{o}(d.ins)} \{a,b,c,d\}\rangle \models e_x$

3.2 Event Choice Function

The previous examples were defined s.t. only one event is triggered at each moment. However, this may not be the case, and for that a reactive language specifies a *operational semantics* which encases two major decisions: 1) in which order should events be responded and 2) how should an event be responded. The former defines the handling order of events in case of conflict, e.g. based on when events have occurred (temporal order), on a priority list, or any other criteria. Then, the latter decision defines the response policy of an ECA-language, i.e. when is an event considered to be responded. E.g., if an event occurs more than once before the system can respond to it, this specifies if such response should be issued only once or equally to the amount of occurrences.

Choosing the appropriate operational semantics depends on the application in mind. E.g., in system monitoring applications there may exist event alarms with higher priority over others and that need to be responded only once per occurrence. Contrarily, in a webstore context it may be important to treat events related to orders with a temporal criteria, and to respond to each occurrence individually. To address these differences $\mathcal{TR}^{ev}$ is parametric on a function *choice*:

Definition 10 (*choice function*)
Let M be an interpretation and π be a path. Then function $choice(M,\pi)$ is defined as follows:

$$choice(M,\pi) = \mathit{firstUnans}(M,\pi,order(M,\pi))$$

As stated, $choice$ defines at each moment what is the next event to be responded. For that, it is based on a $order$ function which sorts events w.r.t. a given criteria, and a function $\mathit{firstUnans}$ which checks what events are unanswered and returns the first one based on this order.

Example 5 (*Ordering-Functions*)
Let $\langle e_1,\ldots,e_n\rangle$ be a sequence of events, π a path and M an interpretation.

Temporal Ending Order $order(M,\pi) = \langle e_1,\ldots,e_n\rangle$ iff $\forall e_i$ s.t. $1 \leq i \leq n$ then $\exists \pi_i$ subpath of π where $M,\pi \models_{\mathcal{TR}} \mathbf{o}(e_i)$ and $\forall e_j$ s.t. $i < j$ then e_j occurs after e_i

Temporal Starting Order $order(M,\pi) = \langle e_1,\ldots,e_n\rangle$ iff $\forall e_i$ s.t. $1 \leq i \leq n$ then $\exists \pi_i$ subpath of π where $M,\pi \models_{\mathcal{TR}} \mathbf{o}(e_i)$ and $\forall e_j$ s.t. $i < j$ then e_j starts before e_i

Priority List Order Let L be a priority list where events are linked with numbers starting in 1, where 1 is the most priority event. $order_L(M,\pi) = \langle e_1,\ldots,e_n\rangle$ iff $\forall e_i\ \exists \pi_i$ subpath of π s.t. $M,\pi_i \models_{\mathcal{TR}} \mathbf{o}(e_i)$ and $\forall e_j$ where $1 \leq i < j \leq n$, π_j is subpath of π and $M,\pi_j \models_{\mathcal{TR}} \mathbf{o}(e_j)$ then $L(e_i) \leq L(e_j)$.

These examples are based on the notion of order of events, which is standardly defined as:

Definition 11 (*Ordering of Events*)
Let e_1, e_2 be events and π a path and M an interpretation. We say that e_2 occurs after e_1 w.r.t. π and M iff there exists π_1, π_2 subpaths of π such that $\pi_1 = \langle D_i,^{O_i}\ldots,^{O_{j-1}} D_j\rangle$, $\pi_2 = \langle D_n,^{O_n}\ldots,^{O_{m-1}} D_m\rangle$, $M,\pi_1 \models \mathbf{o}(e_1)$, $M,\pi_2 \models \mathbf{o}(e_2)$ and $D_j \leq D_m$ w.r.t. the ordering in π. We say that e_1 starts before e_2 w.r.t. π if $D_i \leq D_n$

After exemplifying how events can be ordered, it remains to define the response policy, i.e. what requisites should be imposed w.r.t. the response executions. As illustration, as follows we define a function $\mathit{firstUnans}$ that retrieves the first unanswered event e, given the ordering function above. In this example, if an event occurs more than once before it is responded, then it is sufficient to respond to it once. Other definitions are possible, e.g. where $\mathit{firstUnans}$ requires the execution of a response for every individual occurrence, but these are omitted for lack of space.

Definition 12 (*Answering Choice*)
Let M be an interpretation, π a path, $\langle e_1,\ldots,e_n\rangle$ an event sequence. $\mathit{firstUnans}(M,\pi,\langle e_1,\ldots, e_n\rangle) = e_i$ iff e_i is the first event in $\langle e_1,\ldots,e_n\rangle$ where given π' subpath of π and $M,\pi' \models_{\mathcal{TR}} \mathbf{o}(e)$ then $\neg\exists\pi''$ s.t. π'' is also a subpath of π, π'' is after π' and $M,\pi'' \models \mathbf{r}(e)$.

3.3 Entailment and Properties

As in $\mathcal{TR}$, $\mathcal{TR}^{ev}$ defines two entailments: logical and executional entailment. Logical entailment is defined exactly as in Def. 4 and can be used to reason about properties of transaction and event formulas that hold for *every* possible path of execution. It can also express relations between transactions and occurrences, e.g. to say "event ψ occurs whenever transaction ϕ succeeds".

Contrarily, executional entailment is used to talk about properties of a particular execution path. However, to reason about the execution of transactions over a specific path, care must be taken since, as described above, the satisfaction of a new occurrence in a path may invalidate transaction formulas that were previously true. As such, adding a new rule to a program may make a formula that was previously satisfied in a path π to be false in π.

To deal with a similar behavior, non-monotonic logics rely on the concept of minimal or preferred models: instead of considering all possible models, non-monotonic theories restrict to the most skeptical ones. Likewise, $\mathcal{TR}^{ev}$ uses the minimal models of a program to define entailment, whenever talking about a particular execution of a formula. As usual, minimality is defined by set inclusion on the amount of predicates that an interpretation satisfies, and a minimal model is a model that minimizes the set of formulas that an interpretation satisfies in a path.

Definition 13 (*Minimal Model*)
Let M_1 and M_2 be interpretations. Then $M_1 \leq M_2$ if $\forall\pi$: $M_2(\pi) = \top \vee M_1(\pi) \subseteq M_2(\pi)$
Let ϕ be a $\mathcal{TR}^{ev}$ formula, and P a program. M is a *minimal model* of ϕ (resp. P) if M is a model of ϕ (resp. P) and $M \leq M'$ for every model M' of ϕ (resp. P).

Thus, to know if a formula succeeds in a particular path, we need only to consider the event occurrences *supported* by that path, either because they appear as occurrences in the transition of states, or because they are a necessary consequence of the program's rules given that path. Because of this, executional entailment in $\mathcal{TR}^{ev}$ is defined w.r.t. minimal models (cf. Def. 5).

Definition 14 ($\mathcal{TR}^{ev}$ *Executional Entailment*)
Let P be a program, ϕ a transaction formula and $D_1,^{O_0} \ldots,^{O_n} D_n$ a path. Then statement $P,(D_1,^{O_0} \ldots,^{O_n} D_n) \models \phi$ ($\star$) iff for every minimal model M of P, $M, \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle \models \phi$. $P, D_1- \models \phi$ is said to be true, if there is a path $D_1,^{O_0} \ldots,^{O_n} D_n$ that makes ($\star$) true.

Example 6
Recall program P of example 4 above. Since for every minimal model M_m of P we have $M_m, \langle\{\},^{\mathbf{o}(e_x)} \{\},^{\mathbf{o}(a.ins)} \{a\},^{\mathbf{o}(c.ins)} \{a,c\},^{\mathbf{o}(b.ins)} \{a,b,c\},^{\mathbf{o}(d.ins)} \{a,b,c,d\}\rangle \models e_x$ then we conclude $P, (\{\},^{\mathbf{o}(e_x)} \{\},^{\mathbf{o}(a.ins)} \{a\},^{\mathbf{o}(c.ins)} \{a,c\},^{\mathbf{o}(b.ins)} \{a,b,c\},^{\mathbf{o}(d.ins)} \{a,b,c,d\}) \models e_x$

Interestingly, as in logic programs, formulas satisfied by this entailment have some support.

Lemma 2 (*Support*)
Let P be a program, π a path, ϕ a transaction atom. Then, if $P, \pi \models \phi$ one of the following holds:

1. ϕ is an elementary action and either $\phi \in \mathcal{O}^d(\pi)$ or $\phi \in \mathcal{O}^t(\pi)$;
2. ϕ is the head of a transaction rule in P ($\phi \leftarrow body$) and $P, \pi \models body$;

Proof
Clearly by Definitions 1 and 13 both items are true in all minimal models of program P. As such, it remains to show that every ϕ that holds in a path arises from these. Let's assume that ϕ does not fall in either of the two previous cases. Then, since ϕ is a transaction atom, then by definition of $\mathcal{ETR}$ language, ϕ must either be (1) primitive defined in the oracles but where $\phi \notin \mathcal{O}^d(\pi)$ and $\phi \notin \mathcal{O}^t(\pi)$; or (2) a transaction name that is neither the head of a transaction rule, or that it is the head and $P, \pi \not\models body$; or (3) ϕ appears in the body of a rule whose head is *not* true;

Let's consider each of these cases individually.
(1) if ϕ is an oracle primitive, then ϕ cannot appear in the head of rules in P. Additionally, since the oracles do not make ϕ true in π, then there is no reason for an interpretation that is a minimal model of P to make ϕ true in π. And thus, a minimal model that only respects the oracle primitives and the rules of P does not make ϕ true in π, and thus ϕ is not true in *all* minimal models of P, and $P, \pi \not\models \phi$.

(2) if ϕ is a transaction name that is not the head of a transaction rule, then ϕ cannot be true in any minimal model of P. Moreover, if ϕ appears in the head of a rule where $P, \pi \not\models body$, then it means that at least one minimal models fails to satisfy $body$ in path π. As such, it means that at in that minimal model ϕ does not need to be true from that rule in path π.
(3) The latter case is the trickiest and is the case where for all minimal models M, they must satisfy a rule $head \leftarrow body$ in π and the head is known to be false in all minimal models for path π. However, for a given head to be false in all minimal models, it means that $M, \pi \models head$ is not the case in every minimal model of M. For that to be true, and since such rule exists and $body$ is true in π for all minimal models, then $\neg head$ must be a direct consequence of the program. Since negation cannot appear in the head of rules, and the failure of a primitive oracle does not directly cause failure in the interpretation (cf. Definition1), then this case is impossible. □

As expected, $\mathcal{TR}^{ev}$ extends $\mathcal{TR}$. Precisely, if a program P has no complex event rules, and for every elementary action a defined by the oracles the only rule for $\mathbf{r}(a)$ in P is $\mathbf{r}(a) \leftarrow \texttt{true}$, then executional entailment in $\mathcal{TR}^{ev}$ can be recast in $\mathcal{TR}$. As an immediate corollary of this, it follows that if P is *event-free* then executional entailment in $\mathcal{TR}^{ev}$ and in $\mathcal{TR}$ coincide.

To prove these results, we start by formalizing some auxiliary lemmas and properties.

Definition 15 (*Valid Paths*)
Given a transaction program P, a path $D_1, ^{O_0} \ldots, ^{O_n} D_n$ is said to be valid iff for every occurrence such that $\mathbf{o}(e) \in O_i (0 \leq i \leq n)$ it exists a transaction rule in P of the form $\mathbf{r}(e) \leftarrow \psi$.

To respond to occurrences is of paramount importance in $\mathcal{TR}^{ev}$ in the sense that, not responding to an event occurrence in a path implies transaction formulas to not hold in that path. As an implicit result of this behavior, if an event occurs in a path and this event does not have a response defined, then the event will prevent all transaction formulas to succeed. This can be circumvented by adding the rule $\mathbf{r}(e) \leftarrow true$ whenever the response of event e is not meant to constrain the satisfaction of transactions.

Theorem 2 (*Valid Paths and Satisfaction*)
Let P be a $\mathcal{TR}^{ev}$ program, and ϕ, ψ any two transaction formulas. If $P, D_0, ^{O_1} \ldots, ^{O_n} D_n \models \phi$ then $\forall \mathbf{o}(e) \in O_i (1 \leq i \leq n)$ it exists a rule $\mathbf{r}(e) \leftarrow \psi \in P$

Proof
We prove this property by contradiction. Assume that it exists a path $\langle D_j, ^O D_{j+1} \rangle$ and where $\mathbf{o}(e) = O$ and it does not exist a rule such that $\mathbf{r}(e) \leftarrow \psi$. Also, let M_1 and M_2 be two minimal models of P. By Definition M_1 and M_2 must satisfy $\mathbf{o}(e)$ in that path, but there is no restriction (by the program) on how to satisfy $\mathbf{r}(e)$. However, note that both M_1 and M_2 must satisfy $\mathbf{r}(e)$ otherwise the operator exp will diverge and M_1 and M_2 would not be models. Since both M_1 and M_2 need to satisfy $\mathbf{r}(e)$ in some path, but there is no imposition of the program on how this must be done, we can simply define it for M_1 as follows:

- for any path $\langle D_j, ^O D_{j+1} \rangle$: whenever $\mathbf{o}(e) \in M_1(\langle D_j, ^O D_j \rangle)$ and $\mathbf{r}(e) \notin P$ then $\mathbf{r}(e) \in M_1(\langle D_j, ^O D_j \rangle)$
- for any path $\langle D_j, ^O D_{j+1} \rangle$: whenever $\mathbf{o}(e) \in M_2(\langle D_j, ^O D_j \rangle)$ and $\mathbf{r}(e) \notin P$ then $\mathbf{r}(e) \in M_2(\langle D_k \rangle)$ where D_k is any state except D_j, D_{j+1}.

Clearly, if $\mathbf{r}(e) \notin P$ then $\mathbf{r}(e)$ will never be satisfied in the same path in M_1 as it is in M_2. Since both are minimal models, the convergence of exp operator for a path that contains occurrences

$\mathbf{o}(e)$ of such events will never converge at the same path. As a result for any formula ϕ, ϕ will never be proven by *all* minimal models of P in a path that contains occurrences which response is not defined in the program. $\square$

Based on this last result, we can state that transactions are only executable in paths where all events that occur can be responded.

Theorem 3
Let P be a transaction program and ϕ a transaction formula. Then,
$P, D_1,^{O_0} \ldots,^{O_n} D_n \models \phi$ iff $M, \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle \models \phi$ for all models that are minimal in valid paths

Proof
Immediately from Theorem 2 $\square$

By definition, all primitive actions are also event occurrences. To distinguish the cases where events occur because of explicit events from inferred primitive actions we state the following:

Definition 16
We say that an annotated path π does not have explicit events if for every occurrence O_i in an annotated path π, $O_i \in \mathcal{P}_{\mathcal{O}}$.

Definition 17 (*Event-Free Program*)
We say that P is an event-free program iff P does not contain complex event rules,for all primitive actions $\varphi \in \mathcal{P}_{\mathcal{O}}$ it exists the rule $\mathbf{r}(\varphi) \leftarrow true$ in P and if it does not contain any formula from $\mathcal{P}_e$ in the body of a transaction rule.

Corollary 1
Let P be an event-free $\mathcal{TR}^{ev}$ program, and ϕ an event-free transaction formulas. If $P, D_1,^{O} \ldots,^{O_n} D_n \models \phi$ then $D_1,^{O} \ldots,^{O_n} D_n$ does not have explicit events.

Proof
Special case of Theorem 2 $\square$

Corollary 2
Let P be an event-free program and ϕ an event-free formula.
$P, D_1,^{O_0} \ldots,^{O_n} D_n \models \phi$ iff $M, \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$ for all models that are minimal in paths without explicit events.

Proof
Immediately from Corollary 1 and Theorem 3 $\square$

Lemma 3
Let P be an event-free program and M a minimal model of P. For every path π without explicit events then $\mathtt{exp}_M(\pi) = \pi$.

Proof
Since P is an event-free program, it means that complex event rules do not exist in P and every response of primitive occurrences hold trivially in every path. Thus, if π does not have explicit events and M is minimal, every occurrence that is made true in π is trivially responded. Consequently $\texttt{exp}_M(\pi) = \pi$. $\square$

Lemma 4 (*Minimal Models in $\mathcal{TR}$*)
For all models M of P and for all the minimal models M_M of P the following property is true:

$$M \models_{\mathcal{TR}} \phi \text{ iff } M_M \models_{\mathcal{TR}} \phi$$

Proof
$\Rightarrow$: If $M \models_{\mathcal{TR}} \phi$ then $M_M \models_{\mathcal{TR}} \phi$
Trivially true. If $M \models_{\mathcal{TR}} \phi$ then ϕ is true in *all* models of P including in the minimal ones: M_M.
$\Leftarrow$: If $M_M \models_{\mathcal{TR}} \phi$ then $M \models_{\mathcal{TR}} \phi$
Let us assume that the latter is not true, and so ϕ is true in M_M but not in a given model M_F of the program such that M_F is not a minimal model. Then there must exist a path π such that $\phi \in M_M(\pi)$ but $\phi \notin M_F(\pi)$. However by definition of minimal models M_M is a minimal model if for every path π and every model M of P, $M_M(\pi) \subseteq M(\pi)$. Consequently if $\phi \in M_M(\pi)$ but $\phi \notin M_F(\pi)$ then either M_F is a minimal model (and M_M and M_F are incomparable) or M_M is not a minimal model. Since both hypothesis lead to a contradiction, then we prove that $M_M \models_{\mathcal{TR}} \phi$ then $M \models_{\mathcal{TR}} \phi$. $\square$

Lemma 5
Let P be an event-free program and M be a minimal model of P. If π is does not have explicit events, then:

$$\texttt{exp}_M(\pi) = \pi$$

Proof
Let's assume that $M(\pi) \neq \top$, otherwise the proof is trivial. We know that $\texttt{exp}_M(\pi) = \pi$ if all event occurrences that are true in any subpath of π are responded within π. So let's assume the contrary, i.e. that it exists an event that occurs in a subpath π' of π ($M, \pi' \models \mathbf{o}(e)$) but $\neg\exists\pi''$ subpath of π such that π'' is after π' and $M, \pi'' \models \mathbf{r}(e)$. Since M is a minimal model of P, and P does not contain complex event rules, then $M, \pi' \models \mathbf{o}(e)$ only if $\pi' = \langle D_i, {}^O D_i\rangle$ and $\mathbf{o}(e) \in O$. However, by definition there is no occurrence in any transition of π and consequently $\neg\exists e : M, \pi' \models \mathbf{o}(e)$ $\square$

Next we will define a function s that will allow us to prove the desired correspondence between $\mathcal{TR}^{ev}$ and $\mathcal{TR}$.

Definition 18
We define s as a function from Herbrand path structures to Herbrand path structures defined as follows:

- $\forall \phi, \pi$ such that π is an annotated path, ϕ is an atomic formula and $\phi \notin \mathcal{P}_e$. If $\phi \in M(\pi)$ then $\phi \in s(M(\pi'))$ where π' is obtained by eliminating the annotated transitions of π.
- $\forall e, \pi$ such that π is an annotated path, $e \in \mathcal{P}_e$. If $\mathbf{o}(e) \in M(\pi)$ then $\mathbf{o}(e) \notin s(M(\pi))$

Lemma 6
Let M_1 and M_2 be two interpretations in $\mathcal{TR}^{ev}$

$$M_1 \subseteq M_2 \Rightarrow s(M_1) \subseteq s(M_2)$$

Proof
Immediately from the definition of $s(M)$. $\square$

Definition 19
Let M be an interpretation in $\mathcal{TR}$, then we define M^{-1} to be an interpretation in $\mathcal{TR}^{ev}$ as follows:

- $\forall \pi$ if $p \in M'(\pi)$ then $p \in M^{-1}(\pi)$
- whenever $\exists \mathbf{o}(e)$ such that $D_1,^O D_1$, $\mathbf{o}(e) \in O$ and $e \in \mathcal{P_O}$ then $\mathbf{o}(e) \in M^{-1}, \langle D_1,^O D_1\rangle$ and $\mathbf{r}(e) \in M^{-1}, \langle D_1,^O D_1\rangle$.

Lemma 7
Let M be a herbrand path structure in $\mathcal{TR}$, then $s(M^{-1}) = M$

Proof
Immediately from definition of s. $\square$

Lemma 8
Let P be an event-free transaction program. If M is a minimal model of P in $\mathcal{TR}$ then M^{-1} is also a minimal model of P in $\mathcal{TR}^{ev}$.

Proof
Since for every occurrence $\mathbf{o}(e)$ that M^{-1} makes true in, $\mathbf{r}(e)$ is also true, then $\mathtt{exp}_{M^{-1}}(\pi) = \pi$ for any annotated path π. As a result it is trivial to show that for every transaction formula ϕ and every path π' if $M, \pi_1 \models_{TR} \phi$ then $M^{-1}, \pi_1 \models \phi$, and consequently if M' is a model of P in $\mathcal{TR}$ then M^{-1} is a model of P in $\mathcal{TR}^{ev}$.

It remains to show that M^{-1} is minimal if M is also minimal. So let's assume that it exists a M_{min} such that $M_{min} \subset M^{-1}$. This is equivalent to say that $\exists \phi, \pi$ such that $\phi \in M^{-1}(\pi)$ but $\phi \notin M_{min}(\pi)$. Also, as a result by Lemma 6 it follows that $s(M_{min}) \subseteq s(M^{-1})$ Since M is minimal and $s(M^{-1}) = M$ then $s(M^{-1}) = s(M_{min})$ Let's analyze the possible structure of ϕ

- $\phi \notin \mathcal{P}_e$
  By definition 18 if $\phi \in M^{-1}(\pi)$ but $\phi \notin M_{min}(\pi)$ then $\phi \in s(M^{-1})(\pi)$ but $\phi \notin s(M_{min})(\pi)$. Then $s(M^{-1}) \neq s(M)$ which leads to a contradiction.
- $\phi = \mathbf{o}(e)$ or $\phi = \mathbf{r}(e)$
  Moreover, for all formulas $\mathbf{o}(e)$ such that $\mathbf{o}(e) \in M^{-1}(\pi)$, since π must be $D_1,^O D_1$ and $\mathbf{o}(e) = O$ then by Definition 1 **all** interpretations must also satisfy $\mathbf{o}(e)$ in that path including M_{min}. Furthermore, in order for any given M_{min} to be model, and by definition of an event-free program, for every path π then $M_{min}, \pi \models \mathbf{r}(e)$.

$\square$

Lemma 9
Let P be an event-free program and M a model of P.
If $M \models P$ and $M' = s(M)$ then $M' \models_{\mathcal{TR}} P$

Proof
P is a transaction base, i.e. it is formed by a set of rules with the structure $h \leftarrow \phi$ where ϕ is any transaction formula (event-free) and h is an atomic formula. M is a model of P iff $\forall h \leftarrow b \in P$, $M, \pi \models \phi \leftarrow b$. To prove this, it is sufficient to show that whenever $M', \pi \models_{\mathcal{TR}} \phi$ then $M', \pi \models_{\mathcal{TR}} h$.

Let's assume that $M', \pi \models \phi$ We will prove by induction on the structure of ϕ, note that ϕ cannot be an event-formula since P is event-free.

- **Base Case**: If ϕ is an atomic formula, then $M', \pi \models \phi$ iff $\phi \in M'(\pi)$. Since M is a model of P we know that whenever $M, \pi \models \phi$ then $M, \pi \models h$. By the definition of $\mathcal{TR}$: $M', \pi \models \phi$ iff $\phi \in M'(\pi)$, and thus by Definition 18 we have that $\phi \in M(\pi)$. Moreover by Lemma 3 we know that $\mathtt{exp}_M(\pi') = \pi'$ for every π' that does not have explicit event occurrences. Thus $M, \pi' \models h$ which implies that $h \in M(\pi')$. Since $\mathcal{TR}$ does not have annotated occurrences we know that for every path π valid in $\mathcal{TR}$ it holds that $M, \pi \models_{\mathcal{TR}} h$
- **Serial Conjunction** $\phi = \phi_1 \otimes \phi_2$:
By definition, $M', \pi \models_{\mathcal{TR}} \phi_1 \otimes \phi_2$ iff $M', \pi_1 \models_{\mathcal{TR}} \phi_1$ and $M', \pi_2 \models_{\mathcal{TR}} \phi_2$ and $\pi = \pi_1 \circ \pi_2$. Since $\mathcal{TR}$ does not have any annotations in the paths, we can assume that π does not have explicit occurrences. Then by Lemma 3 we know that $\mathtt{exp}_M(\pi) = \pi$. If this is the case, then it follows that $M, \pi \models \phi_1 \otimes \phi_2$. As a result, $M, \pi \models h$ and thus since $\mathtt{exp}_M(\pi) = \pi$ then $h \in M(\pi)$ and by definition 18 $h \in M'(\pi)$ and thus $M', \pi \models_{\mathcal{TR}} h$.
- **"Classical" Conjunction:** Let's assume this is true for ϕ, i.e. whenever $M', \pi \models_{\mathcal{TR}} \phi$ then $M', \pi \models_{\mathcal{TR}} h$ if $h \leftarrow \phi \in P$. We want to prove that it still holds for $\phi_1 \wedge \phi_2$. So assume $M', \pi \models_{\mathcal{TR}} \phi_1 \wedge \phi_2$ then we know that $M', \pi \models_{\mathcal{TR}} \phi_1$ and $M', \pi \models_{\mathcal{TR}} \phi_2$. By I.H. we can conclude that $M', \pi \models_{\mathcal{TR}} h$.
- **Executional Possibility:** Let's assume this is true for ϕ, i.e. whenever $M', \pi \models_{\mathcal{TR}} \phi$ then $M', \pi \models_{\mathcal{TR}} h$ if $h \leftarrow \phi \in P$. We want to prove that it still holds for $\Diamond\phi$. So assume $M', \pi \models_{\mathcal{TR}} \Diamond\phi$ then π is a 1-path and it exists a π' starting in π s.t. $M, \pi' \models_{\mathcal{TR}} \phi$. Moreover, we know that M is a model of the program and that $M' = s(M)$. Then if ϕ is an atomic formula, then $\phi \in M'(\pi')$ and $M, \pi \models h$ which means that $h \in M(\pi)$ and thus $M, \pi \models_{\mathcal{TR}} h$. Otherwise, if ϕ is not an atomic formula, then we can still decompose it in one of the cases above and try to prove it inductively again. However, the formula must terminate in an atom, and thus we can say that eventually for some π'' starting in π, $M, \pi'' \models \phi$ and thus $M, \pi \models h$, i.e. $h \in M(\pi)$ and thus $M, \pi \models_{\mathcal{TR}} h$.

□

Definition 20
The relation $\subseteq^*$ is a binary relation for any two interpretations M_1, M_2 is defined as follows.

$$M_1 \subseteq^* M_2$$

iff $\forall \pi$ such that π is valid w.r.t. P then $M_1(\pi) \subseteq M_2(\pi)$

The relation $=^*$ is a binary relation for any two interpretations M_1, M_2 is defined as follows.

$$M_1 =^* M_2$$

iff $\forall \pi$ such that π is valid w.r.t. P and $M_1(\pi) = M_2(\pi)$

Lemma 10

$$M_1 \subseteq^* M_2 \Leftrightarrow s(M_1) \subseteq s(M_2)$$

Proof
Immediate by definition as for paths π without explicit events $M(\pi) = s(\pi)$ $\square$

Lemma 11

$$M_1 =^* M_2 \Leftrightarrow s(M_1) = s(M_2)$$

Proof
Immediate by definition as for paths π without explicit events $M(\pi) = s(\pi)$ $\square$

Lemma 12
Let π be a path without explicit events. M is minimal w.r.t. π in $\mathcal{TR}^{ev}$ iff $s(M)$ is minimal w.r.t. π in $\mathcal{TR}$

Proof
Immediate by Lemmas 10 and 11 $\square$

Theorem 4
Let P be an **event-free** transaction program and $\models_{\mathcal{TR}}$ the satisfaction relation specified in (Bonner and Kifer 1993). Then the following property is true for every annotated path $D_1,^{O_0} \ldots,^{O_n} D_n$ without explicit events and every transaction formula ϕ such that $\phi \notin \mathcal{P}_e$:

$$M, D_1,^{O_0} \ldots,^{O_n} D_n \models \phi \text{ iff } s(M), D_1, \ldots, D_n \models_{\mathcal{TR}} \phi$$

Proof
$\Rightarrow$:
Proof by induction on the structure of ϕ. Recall that ϕ is event-free and thus we don't need to consider the event case.

- **Base Case:** Since $\langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$ is without explicit events then by Lemma 5: $\texttt{exp}_M(\langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle) = \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$ and thus we know that: $M, \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle \models \phi$ iff $\phi \in M(\langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle)$. By definition of $s(M)$, since ϕ is not an event, then $\phi \in s(M)(\langle D_1, \ldots, D_n\rangle)$ and consequently, $s(M), \langle D_1, \ldots, D_n\rangle \models_{\mathcal{TR}} \phi$.

- **Serial Conjunction Case:** $\phi = \phi_1 \otimes \phi_2$
  Since $\langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$ does not have explicit events then Lemma 5 tells us that it holds $\mathtt{exp}_M(\langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle) = \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$ and thus $M, \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle \models \phi_1 \otimes \phi_2$ iff $M, \pi_1 \models \phi_1$ and $M, \pi_2 \models \phi_2$ and $\pi_1 \circ \pi_2 = \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$. By applying the induction hypothesis we know that $s(M), \pi_1' \models_{\mathcal{TR}} \phi_1$ and $s(M), \pi_2' \models_{\mathcal{TR}} \phi_2$ for paths π_1' and π_2' that are obtained from π_1 and π_2 by eliminating the annotated transitions. Therefore, since $\pi_1 \circ \pi_2 = \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$ then $\pi_1' \circ \pi_2' = \langle D_1, \ldots, D_n\rangle$. Giving this we can conclude that $s(M), \langle D_1, \ldots, D_n\rangle \models_{\mathcal{TR}} \phi_1 \otimes \phi_2$
- **Negation**: $\phi = \neg\phi$
  So assume that $M, \pi \models \neg\phi$. Then it is not the case that $M, \pi\rangle \models \phi$. Moreover since π does not have explicit events then $\mathtt{exp}_M(\pi) = \pi$. In this case, the satisfaction relation coincide and thus since $s(M)$ makes the same event-free formulas true then, $s(M), \pi \models_{\mathcal{TR}} \phi$.
- **"Classical" Conjunction:** Trivially true by applying the induction hypothesis.
- **Executional Possibility:** Trivially true by applying the induction hypothesis and since $\mathtt{exp}_M(\langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle) = \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$.

$\Leftarrow$:
Proof by induction in the structure of ϕ:

- **Base Case:** $s(M), \langle D_1, \ldots, D_n\rangle \models_{\mathcal{TR}} \phi$ iff $\phi \in s(M)(\langle D_1, \ldots, D_n\rangle)$, then by definition, $\phi \in M(\langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle)$. Since $\langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$ does not have explicit events then $\mathtt{exp}_M(\langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle) = \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$, and thus, as intended, it holds that $M, \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle \models \phi$.
- **Serial Conjunction:** $s(M), \langle D_1, \ldots, D_n\rangle \models_{\mathcal{TR}} \phi_1 \otimes \phi_2$ iff it exists a $\pi_1' \circ \pi_2' = \langle D_1, \ldots, D_n\rangle$ and $s(M)\pi_1' \models_{\mathcal{TR}} \phi_1$ and $s(M)\pi_2' \models_{\mathcal{TR}} \phi_2$. Moreover since $\langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$ does not have explicit events then $\mathtt{exp}_M(\langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle) = \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$. Consequently by applying the induction hypothesis we have that $M, \pi_1 \models \phi_1$, and $M, \pi_2 \models \phi_2$ and thus $M, \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle \models \phi_1 \otimes \phi_2$
- **Negation**: Trivially true since $\langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$ does not have explicit events. Then, by definition $s(M)$ satisfies a formula whenever M also satisfies it. Then since it is true that $\mathtt{exp}_M(\langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle) = \langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$, then the relations coincide.
- **"Classical" Conjunction:** Trivially true by applying the induction hypothesis similar to the serial-conjunction case.
- **Executional Possibility:** Trivially true by applying the induction hypothesis and since $\langle D_1,^{O_0} \ldots,^{O_n} D_n\rangle$ is event-free

$\square$

We can now formulate the comparison with $\mathcal{TR}$ as follows.

Theorem 5 (*Simple Comparison with $\mathcal{TR}$*)
Let P be an **event-free** transaction program and $\models_{\mathcal{TR}}$ the satisfaction relation specified in (Bonner and Kifer 1993). Let $D_1,^{O_0} \ldots,^{O_n} D_n$ be a path and ϕ an event-free transaction formula. Then:

$$P, D_1,^{O_0} \ldots,^{O_n} D_n \models \phi \text{ iff } P, D_1, \ldots, D_n \models_{\mathcal{TR}} \phi$$

Proof
By Lemma 1 we know that ϕ will only be proven in P if $D_1,^{O_0}\ldots,^{O_n} D_n$ does not have explicit events.
$\Rightarrow$:
By Definition, we know that $P, D_1,^{O_0}\ldots,^{O_n} D_n \models \phi$ iff for all minimal models M of P, $M, D_1,^{O_0}\ldots,^{O_n} D_n \models \phi$. Moreover by Theorem 3 we know that it is equivalent to check if $M_1, D_1,^{O_0}\ldots,^{O_n} D_n \models \phi$ for all minimal models M_1 that are only minimal in event-free paths.

By Theorem 4 we know that if M models ϕ in $\mathcal{TR}^{ev}$ in a path, then $s(M)$ also models ϕ in $\mathcal{TR}$ in the same path. By Lemma 12 we know that if M is a minimal model of P w.r.t. event-free paths iff $s(M)$ is also a minimal model of P in $\mathcal{TR}$ w.r.t. the same path.

Since for every minimal model M_1 of P in $\mathcal{TR}$ we know that it exists a M_2 such that M_2 is a minimal model in $\mathcal{TR}^{ev}$ and $s(M_2) = M_1$, then we can conclude that if $M, \langle D_1,^{O_0}\ldots,^{O_n} D_n \models \phi$ for every minimal model M then for every minimal model M_1 of P in $\mathcal{TR}$: $M_1, \langle D_1, \ldots, D_n\rangle \models_{\mathcal{TR}} \phi$

By Lemma 4 we know that if a formula in $\mathcal{TR}$ is satisfied by all minimal models, iff it is also satisfied by the intersection of all models of the program. And therefore by definition, if $P, D_1,^{O_1}\ldots,^{O_{n-1}} D_n \models \phi$ then $P, D_1 \ldots, D_n \models_{\mathcal{TR}} \phi$.

$\Leftarrow$:
By Definition, $P, D_1, \ldots, D_n \models_{\mathcal{TR}} \phi$ iff for all models M of P, $M, D_1, \ldots, D_n \models_{\mathcal{TR}} \phi$. By lemma 4 it is equivalent to check in all minimal models of P in $\mathcal{TR}$.

By Theorem 4 we know that whenever $s(M), D_1, \ldots, D_n \models_{\mathcal{TR}} \phi$ holds then $M, D_1, \ldots, D_n \models_{\mathcal{TR}} \phi$ also holds. Since for every minimal model $M_{\mathcal{TR}}$ of P it exists a minimal model M_1 such that $s(M_1) = M_{\mathcal{TR}}$. Moreover since the minimal models w.r.t. interesting paths coincide for $\mathcal{TR}^{ev}$and $\mathcal{TR}$ then if $s(M), D_1, \ldots, D_n \models_{\mathcal{TR}} \phi$ is true for all minimal models of $\mathcal{TR}$ then $M, D_1, \ldots, D_n \models_{\mathcal{TR}} \phi$ for all models that are minimal w.r.t. event-free paths. And thus by Theorem 3 it follows that ϕ is proven in every minimal model. Therefore it holds that: $P, D_1,^{O_0}\ldots,^{O_n} D_n \models \phi$ $\square$

Based on this we can prove a stronger relation with $\mathcal{TR}$. Similarly to what was done for the previous result we have to start with some basic lemmas as follows.

Definition 21
We define t as a function from interpretations to interpretations defined as follows:

- $\forall \phi, \pi$ such that π is an annotated path, ϕ is an atomic formula that $\phi \notin \mathcal{P}_e$.
$$\text{If } \phi \in M(\pi) \text{ then } \phi \in t(M(\pi^*))$$
- $\forall e \in \mathcal{P}_{events}$ such that $e \in M(\pi_1 \circ \pi_2) \wedge \mathbf{o}(e) \in M(\pi_1) \wedge \mathbf{r}(e) \in M(\pi_2) \wedge \pi = \pi_1 \circ \pi_2$ then $e \in M(\pi^*)$
- Nothing else belongs to $t(M)(\pi)$

where π^* is obtained by eliminating the annotated transitions of π and collapsing the redundant states $\pi_1 \circ \langle D_i, D_i\rangle \circ \pi_2$ into $\pi_1 \circ \langle D_i\rangle \circ \pi_2$.

Lemma 13
Let M_1 and M_2 be interpretations in $\mathcal{TR}^{ev}$
$$M_1 \subseteq M_2 \Rightarrow t(M_1) \subseteq t(M_{2)}$$

Proof
Immediate by definition 21 $\square$

Lemma 14
Let M be an interpretation, and P be a program without complex events. If M is a minimal model of P in $\mathcal{TR}^{ev}$ then $t(M)$ is also a model of P' in $\mathcal{TR}$, where P' is obtained by P substituting all the rules for events e: $\mathbf{r}(e) \leftarrow \mathit{body}$ by $e \leftarrow \mathit{body}$

Proof
$\forall h \leftarrow \phi \in P'$. We need to prove that if $t(M), \pi \models_{\mathcal{TR}} \phi$ then $t(M), \pi \models_{\mathcal{TR}} h$. Let's prove by induction in the structure of ϕ and assume that $t(M), \pi \models_{\mathcal{TR}} \phi$

- **Base Case**: Then $\phi \in t(M)(\pi)$ and by Definition 21 $\phi \in M(\pi)$ and one of the three cases may be true:
  1. $\mathtt{exp}_M(\pi) = \pi$
     And thus $M, \pi \models \phi$. Since M is a model of ϕ then $M, \pi \models h$ and $t(M), \pi \models_{\mathcal{TR}} h$
  2. $\mathtt{exp}_M(\pi) = \pi_1 \wedge \pi_1 \neq \pi$
     And thus since $\phi \in M(\pi)$ and $\mathtt{exp}_M(\pi) = \pi_1$ then $M, \pi_1 \models \phi$ and $h \in M(\pi_1)$. Since complex events are not possible, if h is not an event, it follows immediately that $t(M), \pi \models_{\mathcal{TR}} h$. If h is an event, the only possible case is that $h = \mathbf{r}(e)$ and thus in P' the rule is defined as $e \leftarrow \phi$ and by definition $t(M), \pi \models_{\mathcal{TR}} e$.
  3. $\neg\exists\pi'$ s.t. $\mathtt{exp}_M(\pi) = \pi'$ then $M, \pi \not\models \mathit{body}$ although $\mathit{body} \in M(\pi)$. Moreover, $\neg\exists\pi'$ s.t. $\mathtt{exp}_M(\pi) = \pi'$ only if $\mathbf{o}(e) \in \pi$. Since body is an atomic formula, then it is only true in 1-paths or 2-paths. If π is a 1-path then $\mathtt{exp}_M(\pi) = \pi$ (and therefore we reach a contradiction). If π is a 2-path, since M is minimal $\mathbf{o}(e) \in \pi$ only if $\pi = \langle D,^{\mathbf{o}(e)} D\rangle$. Moreover, in this path since M is minimal and $\mathbf{o}(e)$ cannot appear in the body of any (transaction) rule, then $\mathit{body} \notin M(\pi)$ which also leads to a contradiction.
- **Event Case:** i.e. $\phi = e$
  Since $\mathbf{o}(e)$ and $\mathbf{r}(e)$ cannot appear in the body of a program without complex event rules, it exists a rule $e \leftarrow \mathit{body} \in P$. Moreover, since $e \in t(M)(\pi)$ then $\mathbf{r}(e) \in M(\pi) \wedge \mathbf{o}(e) \in M(\pi_1) \wedge e \in M(\pi_1 \circ \pi)$. Thus $h \in M(\pi_1 \circ \pi)$. Moreover, $\pi_1 \circ \pi$ reduces to π (as $\mathbf{o}(e)$ is only valid in $\langle D_i,^O D_i\rangle$ paths). Now if h is not an event, it follows immediately that $t(M), \pi \models_{\mathcal{TR}} h$. If h is an event, the only possible case is that $h = \mathbf{r}(e)$ and thus in P' the rule is defined as $e \leftarrow \phi$ and by definition $t(M), \pi \models_{\mathcal{TR}} e$.
- **Serial Conjunction Case:** $\phi = \phi_1 \otimes \phi_2$
  Then $t(M), \pi \models_{\mathcal{TR}} \phi_1 \otimes \phi_2$ iff it exists a split $\pi_1 \circ \pi_2$ of π s.t. $t(M), \pi_1 \models_{\mathcal{TR}} \phi_1$ and $t(M), \pi_2 \models_{\mathcal{TR}} \phi_2$. By applying the Induction Hypothesis we know that $M, \pi_1 \models \phi_1$ and $M, \pi_2 \models \phi_2$. Then one of the three cases may be true:
  1. $\mathtt{exp}_M(\pi) = \pi$
     And thus $M, \pi \models \phi_1 \otimes \phi_2$. Consequently since M is a model, we have that $M, \pi \models h$, i.e. $h \in M(\pi)$. By definition it follows immediate that $h \in t(M)(\pi)$ and $t(M), \pi \models_{\mathcal{TR}} h$
  2. $\mathtt{exp}_M(\pi) = \pi_1 \wedge \pi_1 \neq \pi$
     Then, $M, \pi_1 \models h$ and $\mathtt{exp}_M(\pi) = \pi$ and by definition we have that $h \in M(\pi)$. Since h cannot be an event (as the body is a complex formula, it follows that $h \in t(M)(\pi)$ and $t(M), \pi \models_{\mathcal{TR}} h$

3. $\neg\exists\pi'$ s.t. $\texttt{exp}_M(\pi) = \pi'$
   This may never be the case, since $M, \pi_1 \models \phi_1$ and $M, \pi_2 \models \phi_2$ and complex events are not possible.

- The other cases follow immediate by Induction Hypothesis.

$\square$

Lemma 15
Let $t(M)$ be a minimal model in $\mathcal{TR}$. Then M_t^{-1} is an interpretation in $\mathcal{TR}^{ev}$ defined as follows:

1. $\forall\phi \notin \mathcal{P}_e$ of $\mathcal{TR}^{ev}$
   If $\phi \in t(M)(\pi)$ then $\phi \in M_t^{-1}(\pi^*)$
2. $\forall e \in \mathcal{P}_e$
   If $e \in t(M)(\pi)$ then $\mathbf{r}(e) \in M_t^{-1}(\pi^*)$
3. $\forall \mathbf{o}(e)$
   If $\pi_1 = \langle D_i, ^O D_i\rangle \wedge \mathbf{o}(e) \in O$
   (a) If $e \in t(M)(\pi_2) \wedge \pi = \pi_1 \circ \pi_2$
   then $\mathbf{o}(e) \in M_t^{-1}(\pi_1) \wedge e \in M_t^{-1}(\pi^*)$
   (b) If $\neg\exists e.e \in t(M)(\pi_2) \wedge \pi = \pi_1 \circ \pi_2$
   then $\mathbf{o}(e) \in M_t^{-1}(\pi_1^*)$ and $\mathbf{r}(e) \in M_t^{-1}(\pi_1^*)$

and M_t^{-1} is a model of P in $\mathcal{TR}^{ev}$

Proof
Note that $\texttt{exp}_{M_t^{-1}}(\pi)$ converges for every path π. Then it easy to prove, using the method of 14 that M_t^{-1} is still a model of P in $\mathcal{TR}^{ev}$ $\square$

Lemma 16
If $t(M_1) \subset t(M_2) \Rightarrow M_1 \subset M_2$ or M_1 and M_2 are incomparable

Lemma 17

$$M_1 \subseteq^* M_2 \Leftrightarrow t(M_1) \subseteq (M_2)$$

Proof
Immediate by definition 21 $\square$

Lemma 18

$$M_1 =^* M_2 \Leftrightarrow t(M_1) = t(M_2)$$

Proof
Immediate by definition 21 $\square$

Lemma 19
M is minimal w.r.t. valid paths in $\mathcal{TR}^{ev}$ iff $t(M)$ is minimal in $\mathcal{TR}$

Proof
Immediate by Lemmas 10 and 11 $\square$

Definition 22 (*Complex Event-Free Program*)
We say that P is a complex event-free program iff P does not contain complex event rules and for all primitive actions $\varphi \in \mathcal{P}_{\mathcal{O}}$ it exists the rule $\mathbf{r}(\varphi) \leftarrow true$ in P.

Additionally we also say that a transaction formula is event-free if it does not contain any formula from $\mathcal{P}_e$.

Lemma 20
Let P be a program without complex event rules, M be a minimal model of P, ϕ be a transaction formula such that $\phi \neq \mathbf{o}(e)$. If $\phi \in M(\pi) \wedge \exists e, \pi_s.\mathbf{o}(e) \in M(\pi_s)$ for π_s subpath of π then $e \in M(\pi_1)$ for a π_1 superpath of π_s

Proof
By Lemma 2 we know that every formula in $M(\pi)$ must be supported by the program or the path π. If $\phi \in M(\pi)$ and $\mathbf{o}(e) \in \pi_s$ then it means that ϕ is in π only because it derives from $\mathbf{o}(e)$. Since P does not have complex rules, this can only be the case whenever e is present in the interpretation of π. $\square$

Lemma 21
Let P be a complex-event free program. Let M be a minimal model of P. Let ϕ be a transaction formula such that ϕ is not an event occurrence.

$$\phi \in M(\pi) \Rightarrow \mathtt{exp}_M(\pi) = \pi$$

Proof
Trivially true by Lemma 20, since e is always proven (and thus $\mathbf{r}(e)$) whenever $\mathbf{o}(e)$ and ϕ are simultaneous true in a path. $\square$

Theorem 6
Let P be a complex-event free program, and let P' be obtained from P by replacing in P; every event e and every occurrence of $\mathbf{r}(e)$, such that $e \in \mathcal{P}_e$, by a new fluent p_e. Let π be an annotated path and π' be a path obtained from π s.t. for every event occurrence φ in $\pi_1 \circ \langle D,^{\varphi} D\rangle \circ \pi_2$, the path is transformed into $\pi_1 \circ \langle D\rangle \circ \pi_2$. Let $\models_{\mathcal{TR}}$ the satisfaction relation specified in (Bonner and Kifer 1993). Then the following property is true for every transaction formula ϕ:

$$M, \pi \models \phi \text{ iff } t(M), \pi' \models_{\mathcal{TR}} \phi$$

Proof
$\Rightarrow$:
Trivial since Lemma 21 and thus $\mathtt{exp}_M(\pi) = \pi$
$\Leftarrow$:
Trivially true since for every formula $\phi \in t(M)(\pi)$ it must be in $M(\pi')$. Also for every formula $e \in t(M)(\pi)$ it must exist a $\mathbf{o}(e)$ and an $\mathbf{r}(e)$ in π' and π is the simplification of π'. Thus $\mathtt{exp}_M(\pi') = \pi$. $\square$

Finally we can enunciate the desired property as follows. Note that contrarily to the definition of paths presented herein, as defined in the original version, $\mathcal{TR}$ paths do not contain annotations. As such, we prove this property w.r.t. these original paths.

Theorem 7 (*Comparison to $\mathcal{TR}$*)
Let P be a complex-event free program, and let P' be obtained from P by replacing in P every event e and every response $\mathbf{r}(e)$, s.t. $e \in \mathcal{P}_e$, by a new fluent p_e. Let π be an annotated path and π' be a path obtained from π removing the annotations and for every event occurrence φ in π s.t.$\pi = \pi_1 \circ \langle D,^{\varphi} D\rangle \circ \pi_2$, then $\pi' = \pi_1 \circ \langle D\rangle \circ \pi_2$. Then for every transaction formula ϕ:

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

Proof
$P, \pi \models \phi$ iff $M, \pi \models \phi$ for all minimal models of P. Since from Theorem 6 the models coincide then the result comes immediately from Theorem 6. Similarly we can say that $P', \pi' \models_{\mathcal{TR}} \phi$ iff $t(M), \pi \models_{\mathcal{TR}} \phi$, and thus the reverse is also true. □

4 Discussion and Related Work

Several solutions exist to reason about complex events and execute (trans)actions. Complex event processing (CEP) systems as (Adaikkalavan and Chakravarthy 2004; Wu et al. 2006) can reason very efficiently with large streams of data and detect (complex) events. These support a rich specification of events based on event pattern rules combining atomic events with some temporal constructs. As shown in Theorem 1, $\mathcal{TR}$ and $\mathcal{TR}^{ev}$ can express most event patterns of SNOOP, failing only to translate the expressions that require the explicit specification of time. ETALIS (Anicic et al. 2012) CEP system even uses $\mathcal{TR}$'s syntax and connectives, although abandoning $\mathcal{TR}$'s model theory and providing a different satisfaction definition. However, contrarily to $\mathcal{TR}^{ev}$, CEP systems do not deal with the execution of actions in reaction to the events detected.

To reason about (trans)actions, solutions like the Situation Calculus, Event Calculus, Action Languages, etc. are popular for their ability to model, very expressively, the direct and indirect effects of actions in KBs. Motivated by the success of relational and deductive databases, several extensions of these languages like (Baral et al. 1997; Bertossi et al. 1998) have been proposed to reason about the effects of a subset of actions that follows a transactional behavior and, some of these solutions can also reason about events that behave similarly to database triggers. However, as in database triggers, these events are restricted to detect simple actions like “on insert/delete” and thus have a very limited expressivity that fails to encode complex events, as defined in CEP systems and in $\mathcal{TR}^{ev}$. Further, although the examples herein are all based on a relational oracle, nothing prevents $\mathcal{TR}^{ev}$ to use alternative KBs and subsequently primitive actions. E.g., with an Action Language oracle as defined in (Gomes and Alferes 2013), $\mathcal{TR}^{ev}$ could model reactive transactions where atomic events are the actions defined in a Action Language transition relation.

To simultaneously reason about actions and complex events, ECA languages (Alferes et al. 2011; Bry et al. 2006; Chomicki et al. 2003) are the standard paradigm, following the syntax “on *event* if *condition* do *action*”. These define what are the actions that must be executed based on the occurrence of complex events as well as the effects of these actions. ECA languages normally do not allow the action component of the language to be defined as a transaction, and when they do, they lack from a declarative semantics as (Papamarkos et al. 2006); or they are based on active databases and can only detect atomic events defined as insertions/deletes (Zaniolo 1995; Lausen et al. 1998). Contrarily, $\mathcal{TR}^{ev}$ can define complex events and since it is parametric on a database and transition oracle, atomic events are arbitrary. In $\mathcal{TR}^{ev}$ ECA-rules can be written as:

$$\begin{array}{l}\mathbf{r}(event) \leftarrow \Diamond cond \otimes action \\ \mathbf{r}(event) \leftarrow \neg\Diamond cond\end{array} \quad (3) \qquad \begin{array}{l}\mathbf{r}(event) \leftarrow rule_1 \otimes rule_2 \otimes \ldots \otimes rule_n \\ \text{Assuming that a rule is defined as follows:} \\ rule_i \leftarrow \Diamond cond_i \otimes act_i \\ rule_i \leftarrow \neg\Diamond cond_i\end{array} \quad (4)$$

where the (3) and (4) define the different consumption policies of an ECA. Viz. (3) is the case where for one event occurrence only one ECA-rule is fired, while (4) fires all its ECA-rules.

$\mathcal{TR}^{ev}$ can also be compared with (Chomicki et al. 2003) which proposes a policy description language where policies are formulated as sets of ECA rules and conflicts between policies are captured by logic programs. This work is very interesting as it also provides properties on the execution of the actions, and transaction-like actions can be achieved if the user provides the correct specification for conflict rules. Yet, only a relaxed model of transactions can be achieved and it requires a complete low-level specification of the transaction conflicts by the user.

In the context of multi-agent systems, (Kowalski and Sadri 2012; Costantini and Gasperis 2012) propose logic programming languages that react and execute actions in response to complex events. Unfortunately, these actions fail to follow any kind of transaction model.

$\mathcal{TR}^{ev}$ is a logic programming like language integrating complex events rules with reactive rules that execute transactions. It is parametric on a pair of oracles and on a function $choice$ that picks the event to be responded at a given time. $\mathcal{TR}^{ev}$ gives an important contribution to model transactional behavior in arbitrary reactive systems, and which is elegantly expressed in a non-monotonic manner. With it, one is able to talk about what properties (or fluents) hold after the execution of a reactive transaction, but also, *how* and in which paths such a reactive transaction can succeed. The latter is $\mathcal{TR}^{ev}$'s main innovation: to reason about the correct execution paths of arbitrary transactions that react to internal and external events. We have also defined a procedure to execute these reactive transaction, which is built upon the complex event detection algorithm of ETALIS and the execution algorithm of $\mathcal{TR}$, but it is omitted for lack of space.

References

ADAIKKALAVAN, R. AND CHAKRAVARTHY, S. 2004. Formalization and detection of events over a sliding window in active databases using interval-based semantics. In *ADBIS*. 241–256.

ADAIKKALAVAN, R. AND CHAKRAVARTHY, S. 2006. Snoopib: Interval-based event specification and detection for active databases. *Data Knowl. Eng. 59,* 1, 139–165.

ALFERES, J. J., BANTI, F., AND BROGI, A. 2011. Evolving reactive logic programs. *Intelligenza Artificiale 5,* 1, 77–81.

ANICIC, D., RUDOLPH, S., FODOR, P., AND STOJANOVIC, N. 2012. Stream reasoning and complex event processing in etalis. *Semantic Web 3,* 4, 397–407.

BAILEY, J., DONG, G., AND RAMAMOHANARAO, K. 2004. On the decidability of the termination problem of active database systems. *Theor. Comput. Sci. 311,* 1-3, 389–437.

BARAL, C., LOBO, J., AND TRAJCEVSKI, G. 1997. Formal characterizations of active databases: Part ii. In *DOOD*. LNCS, vol. 1341. Springer, 247–264.

BERTOSSI, L. E., PINTO, J., AND VALDIVIA, R. 1998. Specifying active databases in the situation calculus. In *SCCC*. IEEE Computer Society, 32–39.

BONNER, A. J. AND KIFER, M. 1993. Transaction logic programming. In *ICLP*. 257–279.

BONNER, A. J. AND KIFER, M. 1998. Results on reasoning about updates in transaction logic. In *Transactions and Change in Logic Databases*. 166–196.

BONNER, A. J., KIFER, M., AND CONSENS, M. P. 1993. Database programming in transaction logic. In *DBPL*. 309–337.

BRY, F., ECKERT, M., AND PATRANJAN, P.-L. 2006. Reactivity on the web: Paradigms and applications of the language xchange. *J. Web Eng. 5,* 1, 3–24.

CHOMICKI, J., LOBO, J., AND NAQVI, S. A. 2003. Conflict resolution using logic programming. *IEEE Trans. Knowl. Data Eng. 15,* 1, 244–249.

COSTANTINI, S. AND GASPERIS, G. D. 2012. Complex reactivity with preferences in rule-based agents. In *RuleML*. 167–181.

GOMES, A. S. AND ALFERES, J. J. 2013. External transaction logic with automatic compensations. In *CLIMA*. Springer, 239–255.

KOWALSKI, R. A. AND SADRI, F. 2012. A logic-based framework for reactive systems. In *RuleML*. 1–15.

LAUSEN, G., LUDÄSCHER, B., AND MAY, W. 1998. On active deductive databases: The statelog approach. In *Transactions and Change in Logic Databases*. 69–106.

PAPAMARKOS, G., POULOVASSILIS, A., AND WOOD, P. T. 2006. Event-condition-action rules on rdf metadata in p2p environments. *Comp. Networks 50,* 10, 1513–1532.

WU, E., DIAO, Y., AND RIZVI, S. 2006. High-performance complex event processing over streams. In *SIGMOD Conference*. ACM, 407–418.

ZANIOLO, C. 1995. Active database rules with transaction-conscious stable-model semantics. In *DOOD*. 55–72.