O USO DA LÓGICA DE TRANSAÇÕES EM PLANEJAMENTO[1]

Flavio Tonidandel e Márcio Rillo
Universidade de São Paulo
Escola Politécnica – Departamento de Engenharia Eletrônica
Av. Prof. Luciano Gualberto, 158, trav 3 – São Paulo – SP – Brasil
CEP 05508-900 – e-mail: flavio@clarke.lac.usp.br ; rillo@lsi.usp.br

Abstract: Many logical theories used to formalize planning systems don't allow operations with databases that represent knowledge bases, or don't have any inference rule to allow the computational implementation. Instead of that, Transaction Logic, with its restrictive version called serial-Horn, allows a computational implementation of its theory in PROLOG without any semantic lose and has a semantic equivalence with elements of a planning system, as proved by this work[1].

1 INTRODUÇÃO

Um dos grandes desafios da Inteligência Artificial é desenvolver sistemas inteligentes capazes de serem aplicados nos mais diversos domínios e que apresentem soluções corretas para os mais diversos problemas.

Isto pode ser alcançado através do uso de uma teoria lógica na formalização da base de conhecimento do sistema, de modo a torná-lo independente do domínio e correto com relação a um problema proposto. É necessário, no entanto, uma prova da equivalência semântica entre a teoria e os componentes da base de conhecimento do sistema.

A sub-área da Inteligência Artificial chamada Planejamento ("Planning"), por desenvolver sistemas capazes de gerar planos (seqüência de ações) necessários para alcançar um determinado objetivo, também necessita de um estudo de formalização baseado em alguma teoria lógica que seja capaz de traduzir a semântica de seus componentes, tais como ações, estados e planos.

A formalização de um sistema se torna importante devido ao fato de ser necessário conhecer seus limites, seus procedimentos fundamentais e sua capacidade de descrever o domínio de aplicação. Os sistemas não construídos sob uma formalização teórica rigorosa não possibilitam o conhecimento prévio de seu comportamento diante situações de domínios além daquelas que foram previamente testadas.

Entretanto, não basta apenas a escolha de uma teoria formal que possua uma semântica equivalente à base de conhecimento, mas também que possua capacidade de ser implementado computacionalmente, possibilitando a aplicação prática do sistema formalizado.

Muitas teorias lógicas foram propostas com tal finalidade, mas a maioria falha ou no tratamento de atualizações em uma base de conhecimento ou não possui regras de inferência capazes de traduzirem a teoria em linguagem computacional. Como a lógica temporal utilizada nos sistemas DEVISER (VERE, 1983) e TLPlan (BACCHUS, KABANZA, 1996) e a lógica LLP ("Logical Language for Planning") utilizada pelos sistemas MRL (KOEHLER, 1996) e PHI (KOEHLER, 1996), que possuem uma difícil implementação computacional sem a perda de suas semânticas.

Dentre as teorias utilizadas na modelagem de ações, se destacam o cálculo situacional ("situation calculus – SITCAL") (McCARTHY, HAYES, 1990) e o cálculo de eventos ("event calculus") (KOWALSKI, SERGOT, 1996), ambos baseados na lógica de primeira ordem, primeiramente utilizada pelo sistema de planejamento STRIPS (FIKES, NILSSON, 1971). Embora a lógica de primeira ordem possua uma sintaxe de simples compreensão, ela não trabalha com raciocínio não-monotônico, além de não possuir um tratamento de base de dados.

Com o propósito de evitar os problemas acima descritos decorrentes da aplicação das diversas lógicas aos sistemas de planejamento, este artigo tem por objetivo introduzir o uso da Lógica de Transações (BONNER, KIFER, 1995), abreviada TR, como a teoria formal capaz de fornecer uma modelagem dos componentes de um sistema de planejamento que vise manter sua semântica e possibilite sua implementação computacional.

A TR possui regras de inferência capazes de torná-la computacionalmente implementável, além de uma sintaxe clara e simples (como a lógica de primeira ordem), da capacidade de modelar atividade seqüencial (como a lógica temporal) e também da semântica baseada em caminhos de estados, de forma similar às lógica modais, tal como a lógica LLP. A Lógica de Transações apresenta uma similaridade semântica com os componentes de um sistema de planejamento, que será analisada, desenvolvida e provada por este artigo.

[1] Este trabalho tem o apoio financeiro da FAPESP – Fundação de Amparo à Pesquisa do Estado de São Paulo.

2 LÓGICA DE TRANSAÇÕES

A Lógica de Transações (TR) é uma lógica proposta por BONNER E KIFER (1995) que especifica formalmente fenômenos de atualização de base de dados, caminhos de execução e procedimentos de transições seqüenciais. Em planejamento estas características são comuns, como observado por RILLO (1994), pois um planejador pode ser visto como um processo que, após executar uma seqüência de ações, atualizará o estado do mundo.

A TR é uma extensão da lógica de primeira ordem com a introdução de um novo operador chamado conjunção serial ($\otimes$). Este operador representa uma transação, onde $\alpha \otimes \beta$ significa: "primeiro execute α, e então execute β".

Para descrever uma transação é considerada a seguinte notação: $P,D_0,...,D_n \models \phi$, onde P é chamado de base de transações, que é um conjunto de fórmulas em TR, assim como é ϕ. No entanto, cada D_i é um conjunto de fórmulas de primeira ordem, onde cada uma delas é denominada de literal, que representa a constituição da base de dados. Intuitivamente, P é um conjunto de transações descritas em TR, ϕ é a invocação de algumas dessas transações e $D_0,...,D_n$ é a seqüência da base de dados representando todos os estados da execução de ϕ.

Por exemplo, chamando a transação ϕ, a base de dados vai do estado inicial D_0 ao estado final D_n. Entretanto, se ϕ for somente uma consulta, ou seja, não modificar a base de dados, não haverá caminho de execução e a representação se dará apenas pelo estado em que ϕ é verdadeiro: $P,D \models \phi$.

Em TR, uma base de dados é definida e controlada por oráculos. O estado da base de dados é definido pelo oráculo de dados O^d, sendo que para cada identificador de estados, *i*, $O^d(i)$ é um mapeamento de i para um conjunto de fórmulas de primeira ordem que representam a verdade sobre o estado. Dependendo do domínio de aplicação, diferentes oráculos de dados podem ser especificados. Neste artigo será utilizado o oráculo relacional, onde D é um conjunto de fórmulas atômicas sem variáveis e $O^d(D) = D$. Uma consulta também é controlada pelo oráculo de dados O^d, através do predicado qry(_). Por exemplo, considere o literal *nochão* que está presente no estado D_k. Então, *nochão* $\in O^d(D_k)$ e $P,D_k \models$ qry(*nochão)*.

A TR também trabalha com atualizações e a especificação de transições é feita através do oráculo de transição O^t, que é uma função de mapeamento entre pares de estados e um conjunto de fórmulas atômicas. Neste artigo, uma transição será definida através dos predicados ins(_) e del(_). Por exemplo: ins(c) e del(d) são formalmente executados: se ins(c) $\in O^t(D,D+\{c\})$ e del(d) $\in O^t(D,D-\{d\})$, então $P,D,D+\{c\}, D+\{c\}-\{d\} \models ins(c) \otimes del(d)$. Com estas definições, o seguinte lema é provado (BONNER, KIFER, 1995):

Lema 2.1 : Para qualquer Base de Transações P, qualquer seqüência de estados da base de dados $D_0,...,D_n$ e quaisquer fórmulas de transações α e β, as seguintes afirmações são verdades:

1. Se $P, D_0,...,D_i \models \alpha$ e $P,D_i,...,D_n \models \beta$ então $P,D_0,...,D_n \models \alpha\otimes\beta$.
2. Se $\alpha\leftarrow\beta$ está em P e $P,D_0,...,D_n \models \beta$ então $P,D_0,...,D_n \models \alpha$.
3. Se $O^t(D_0,D_1) \models^c \alpha$ então $P,D_0,D_1 \models \alpha$; onde α é uma fórmula de primeira ordem.
4. Se $O^d(D_0) \models^c \alpha$ então $P,D_0 \models\alpha$; onde α é uma fórmula de primeira ordem.

2.1 Teoria de Provas

Diferentemente de outras lógicas, a semântica da Lógica de Transações é baseada em caminhos de estados, e não em arcos entre estados como nas lógicas modais. Isto quer dizer que, na TR, uma seqüência de estados é estabelecida através do caminho de estados gerado pela execução de alguma transação, i.e.:$P,D_1,...,D_n \models \phi$; onde ϕ é uma transação, e $<D_1,...,D_n>$ é o caminho de estados.

A TR tem uma versão mais restrita, chamada de Horn-serial, que possui uma teoria de provas corretas e completas pela definição de um axioma e regras de inferência. Esta versão restrita usa o operador conjunção serial ($\otimes$) e possui as mesmas características das cláusulas Horn na lógica de primeira ordem. Consiste ainda de uma base de transações P, que é um conjunto de fórmulas Horn-serial e de uma base de dados D, que é um conjunto de fórmulas de primeira ordem.

Com esta versão restrita é possível fornecer regras computáveis à teoria formal. Para isto, são definidas três regras de inferência que operam com deduções para consultas, atualizações e invocações de regras na base de transações P, as quais são necessárias para implementar a dedução em TR e o caminho execucional (BONNER, KIFER, 1995).

A implementação das regras de inferência pode ser feita em linguagem PROLOG. O trabalho de SANTOS (1997) prova a equivalência semântica da versão Horn-serial, com o uso dos predicados ins(_) e del(_), com a semântica de sua implementação em PROLOG. É demonstrado que a representação de um sistema descrito em versão Horn-serial da TR é traduzido em um, e somente um, programa em PROLOG, através do uso de uma função de tradução específica e sem perda da estrutura semântica da teoria.

3 LÓGICA DE TRANSAÇÕES PARA PLANEJAMENTO

Para o uso da Lógica de Transações em planejamento, uma análise das fórmulas em TR é necessária, com o intuito de promover uma aproximação semântica entre a TR e os componentes de um sistema de planejamento, ou seja, planos, estados e ações.

No entanto, além das definições e do lema apresentado na seção anterior, faz-se necessário estabelecer as seguintes definições:

Definição 3.1: *Elem_ins(ϕ) é o conjunto de todos os elementos dos predicados ins(_) existentes em uma fórmula ϕ em TR.*

Definição 3.2: *Elem_del(ϕ) é o conjunto de todos os elementos dos predicados del(_) existentes em uma fórmula ϕ em TR.*

Com as definições 3.1 e 3.2 é possível estabelecer conjuntos com os elementos dos predicados ins(_) e del(_) de uma fórmula em TR. Por exemplo:

ϕ = del(a) $\otimes$ ins(b) $\otimes$ del(c) $\otimes$ ins(d) $\otimes$ ins(e).

Assim,

Elem_del(ϕ) = {a, c}; e
Elem_ins(ϕ)={b, d, e}.

Definição 3.3: *Uma fórmula ϕ em TR, tal que $P,D_0... \models \phi$ é dita independente se e somente se:*

1. *$Elem\_del(\phi) \cap Elem\_ins(\phi) = \{\}$;*
2. *$Elem\_del(\phi) \subseteq D_0$;*
3. *$Elem\_ins(\phi) \cap D_0 = \{\}$.*

A definição 3.3 descreve uma fórmula em TR dita independente. Isso quer dizer que todos os elementos removidos pelos predicados del(_) não podem ser inseridos por predicados ins(_) na mesma fórmula e vice-versa. Para ser independente, uma fórmula ainda deve possuir somente predicados del(_) cujos elementos já pertençam ao estado inicial da base de dados D_0 e possuir somente predicados ins(_) cujos elementos não pertençam a D_0.

A limitação de uma fórmula ser independente é necessária para as definições e provas seguintes. Assim, com a teoria da TR descrita anteriormente e a definição 3.3, pode-se estabelecer o seguinte teorema:

Teorema 3.4: *Sendo uma fórmula independente ψ em TR tal que $P,D_0...D_n \models \psi$, então pode-se afirmar que $D_n = D_0 + Elem\_ins(\psi) - Elem\_del(\psi)$.*

Prova: *Apresentada em (TONIDANDEL,1999)*

Com o teorema anterior, pode-se definir uma função de diferença de conjuntos que servirá para estabelecer o que muda de uma base de dados para outra em TR.

Definição 3.5: *Sendo X e Y conjunto de elementos, define-se uma função de diferença como sendo $Diff(X,Y) = X - (X \cap Y)$.*

Com o uso dos conjuntos de elementos formados por cada base de dados de um caminho em TR, pode-se instituir o seguinte teorema:

Teorema 3.6: *Sendo ψ uma fórmula independente em TR tal que $P,D_0..D_n \models \psi$, as seguintes afirmações são verdades:*

1. *$Diff(D_0,D_n) = Elem\_del(\psi)$.*
2. *$Diff(D_n,D_0) = Elem\_ins(\psi)$.*

Prova: *Apresentada em (TONIDANDEL, 1999).*

Assim, sendo ψ uma fórmula independente, pode-se determinar pela função diferença quais são os elementos inseridos e quais são os elementos retirados pela fórmula, analisando somente o estado inicial da base de dados (D_0) e o estado final (D_n).

Para uma análise das fórmulas da TR, é possível fazer a seguinte definição:

Definição 3.7 (Função Consulta-Estado) *Um função Consulta-Estado de uma fórmula ψ em TR, descrita como $\Delta(\psi)$, é definida como uma função que retorna o conjunto de todos os elementos dos predicados qry(_) presentes em ψ.*

Desta forma é fácil verificar que:

Teorema 3.8: *Se ψ for uma fórmula em TR tal que $P,D \models \psi$, então as seguintes afirmações são verdades:*

1. *$P, \Delta(\psi) \models \psi$*
2. *Se $P,D \models \psi$ então $\Delta(\psi) \subseteq O^d(D)$.*

Prova: *Apresentada em (TONIDANDEL,1999)*

A definição da função Consulta-Estado é importante para o estudo teórico das consultas em uma base de dados que será detalhado mais adiante.

Para aplicar a TR em planejamento é necessário um estudo da possibilidade que a lógica oferece em descrever ações e planos. Tais definições serão importantes para traçar a equivalência entre a semântica da TR e a semântica dos componentes de um sistema de planejamento genérico, que são os planos, as ações e os estados do mundo. Embora a Lógica de Transações já tenha sido aplicada na formalização de ações (SANTOS, 1996) e (SANTOS, RILLO, 1997), não houve a preocupação em estudar a equivalência semântica, que é indispensável tanto para verificar a viabilidade da aplicação da TR em planejamento, quanto para definir a forma de modelar-se os estados, as ações e os planos dentro da estrutura formal da lógica.

Sendo assim, primeiramente é preciso estabelecer uma semântica genérica para os componentes dos sistemas de planejamento. Uma instância de problema de planejamento, de modo genérico, pode ser caracterizada pela seguinte tripla:

$$\psi_P = <\boldsymbol{S}_0, \boldsymbol{G}, \boldsymbol{A}>$$

onde $\boldsymbol{S}_0$ é o estado inicial, $\boldsymbol{G}$ é o objetivo e $\boldsymbol{A}$ é o conjunto de ações do domínio.

Na tentativa de modelar os componentes de um sistema de planejamento, pode-se fazer as seguintes considerações: que opere com suposição de mundo fechado ("Closed-World Assumption"), que possua operadores STRIPS (FIKES, NILSSON, 1971) e que trabalhe em ambientes estáticos aonde somente o agente pode transformá-lo (suposição STRIPS).

Essas considerações são baseadas na semântica do sistema STRIPS (LIFSCHITZ, 1987) e foram escolhidas por serem simples e estarem presentes em muitos sistemas de planejamento, tanto nos chamados clássicos, quanto nos atuais.

Os sistemas de planejamento, em sua grande maioria, dado um problema de planejamento ψ_P, procuram encontrar automaticamente uma seqüência de ações que transforme o estado inicial S_0 em um estado final Sn de forma que o objetivo G seja satisfeito.

O modelo de estado, em vários sistemas, é composto por um conjunto de literais que representa a verdade do mundo naquele instante. Analogamente, o objetivo G também é um conjunto de literais que representa a verdade que se deseja obter ao final do planejamento. A maioria dos sistemas com operadores STRIPS, por sua vez, modifica o estado através de inserções e remoções de literais, de forma a encontrar um conjunto dos mesmos no qual G esteja contido.

Pode-se modelar os componentes de um sistema de planejamento em teoria de conjuntos, a seguir:

Definição 3.9: *Um estado (W) de planejamento é representado por um conjunto de literais que identificam todos os elementos que são verdades do mundo. Onde cada $w \in W$ é chamado de literal. Uma ação em planejamento, com o modelo de operador STRIPS, possui:*

- *Pré-condição (PRE): um conjunto de literais que determinam quando a ação pode ser aplicada em um certo estado W.*

- *Lista de remoções (DL) ("delete list"): um conjunto de literais que serão eliminados do Estado W.*
- *Lista de inserções (IL) ("insert list"): um conjunto de literais que serão adicionados ao Estado W.*
- *Pós-condição (POS): um conjunto de literais que determinam as verdades do mundo que foram adicionadas após a ação ser aplicada*

Um plano em planejamento é caracterizado como sendo uma seqüência de ações.

Com a definição anterior, pode-se estabelecer que o estado inicial $\boldsymbol{S}_0$ será representado por W_0 e o estado final ***Sn*** será representado por Wn. Assim, para modelar a semântica de cada componente:

Definição 3.10 (Semântica de uma ação) *Uma ação é definida como sendo um conjunto α tal que α = PRE +DL + IL + POS. Assim, o comportamento de uma ação dar-se-á na seguinte forma: se PRE $\subseteq$ Wo, então Wn = Wo - DL + IL e POS $\subseteq$ Wn.*

A definição 3.10 diz que a pré-condição (PRE) de uma ação deve ser satisfeita pelo estado inicial (Wo) para que esta possa ser aplicada, ou seja, os elementos de PRE devem pertencer ao conjunto Wo, e, ao ser aplicada, a ação altera o conjunto Wo formando Wn através de uma operação de conjunto Wn=Wo-DL+IL. Após isso, a ação só torna-se verdadeira se os efeitos esperados representados por POS estiverem presentes em Wn.

Definição 3.11 (Semântica de um plano) *Um plano é definido como sendo uma união seqüencial de ações, representado por $\delta = \alpha_1, \alpha_2, \ldots, \alpha_n$ e que apresenta o seguinte comportamento:*

$PRE_1 \subseteq W_0$, tem-se $W_1 = W_0 - DL_1 + IL_1$ e $POS_1 \subseteq W_1$ para α_1
$PRE_2 \subseteq W_1$, tem-se $W_2 = W_1 - DL_2 + IL_2$ e $POS_2 \subseteq W_2$ para α_2
. . .
$PRE_n \subseteq W_{n-1}$, tem-se $W_n = W_{n-1} - DL_n + IL_n$ e $POS_n \subseteq W_n$ para α_n

De tal forma que que o resultado final do plano (Wn) satisfaça G, ou seja, $G \subseteq Wn$.

A definição 3.11 garante a seqüência de ações que caracterizam um plano, onde o estado final de uma ação é o estado inicial da próxima. A definição garante ainda a correção do plano, pois este deve ser tal que $G \subseteq$ Wn.

Definida a semântica de estados do mundo, de ações e de planos, que são os componentes de um sistema de planejamento genérico, pode-se mostrar então a equivalência existente entre os componentes de um sistema de planejamento modelado em teoria de conjuntos e a Lógica de Transações, mas antes é preciso definir igualdade entre conjuntos:

Definição 3.12.: (Igualdade Semântica entre Conjuntos) *Um conjunto C_1 é igual a outro conjunto C_2, se e somente se, cada elemento $x_i \in C_1$ para i=1...n, existir uma correspondência semântica de um-para-um com cada elemento de C_2, ou seja, existindo uma função de C_1 em C_2, existe também a função inversa de C_2 em C_1 da seguinte forma: Se $f: C_1 \rightarrow C_2$ e $f^{-1}: C_2 \rightarrow C_1$, então $C_1=C_2$.*

Ou seja, o conjunto C_1 possui n elementos que semanticamente eqüivalem aos n elementos de C_2. Considere para tanto que dizer que um elemento de um conjunto A é semanticamente equivalente a um elemento do conjunto B, é afirmar que existe uma função aonde o elemento do conjunto B é imagem do elemento do conjunto A e que existe a função inversa aonde o elemento do conjunto A é imagem do elemento do conjunto B.

Com a definição acima e a modelagem de planejamento estabelecido em teoria de conjuntos pode-se definir as equivalências descritas a seguir.

3.1 Equivalência entre Estado (W) em Planejamento e Base de Dados (D) em TR

Para estabelecer uma equivalência entre a semântica da TR e a semântica de um estado em planejamento, considera-se, por simplicidade, o Oráculo Relacional, que define a base de dados como um conjunto de fórmulas atômicas sem variáveis livres de primeira ordem, onde, semanticamente, $O^d(D)=D$.

Assim, a TR possui um conjunto de literais definidas por D e que pode ser usado e definido como sendo o conjunto de literais W em planejamento. Isso só será possível se existir uma relação semântica de um para um entre os literais em W e os literais em D.

Teorema 3.13 (Equivalência entre Estados) *Se para cada literal $fi \in W$ existir um e somente um literal $li \in D$ tal que $\forall i$, fi possui a mesma semântica de li, então W=D.*

Prova: *Sendo que para cada $fi \in W$ existe um e somente um $li \in D$ tal que $\forall i$, fi possui a mesma semântica de li, estabelece-se pela definição 3.12: W=D.* ■

3.2 Equivalência entre Ações (α) em Planejamento e Fórmulas ϕ em TR

Em TR, uma fórmula Horn-serial pode possuir consultas e atualizações na base de dados D. Pelo Teorema 3.6, o conjunto de elementos dos predicados ins(_) e dos predicados del(_) representam o conjunto de mudanças realizadas no estado D_0 para alcançar o estado D_n, em um caminho $\Pi = <D_0...D_n>$, desde que a fórmula Horn-serial responsável por tais mudanças seja independente.

Uma ação possui exatamente a característica de uma fórmula independente em TR. Por ser elementar deve transformar o mínimo possível o estado a que está sendo aplicado, ou seja, nenhum elemento a ser inserido será removido pela própria ação. Seus efeitos também devem ser elementares, pois só poderão remover os elementos que já pertençam ao estado inicial e inserir somente aqueles que não estejam presentes.

Desta forma, pode-se escrever para uma fórmula independente ϕ a seguinte estrutura: $\phi \leftarrow$ PreQry $\otimes$ Upd $\otimes$ PosQry , tal que $P,D_0 \ldots D_n \models \phi$. Onde PreQry e PosQry são conjuntos de consultas à base de dados e Upd é uma fórmula em TR que contém em sua composição somente predicados em concordância com o Oráculo de Transições.

Lema 3.14 : *Para uma fórmula independente ϕ em TR com a seguinte estrutura: $\phi \leftarrow$ PreQry $\otimes$ Upd $\otimes$ PosQry , tal que $P,D_0...D_n \models \phi$; as seguintes afirmações são verdades:*

1. *$P,D_0 \models$ PreQry*
2. *$D_n = D_0 +$ Elem_ins (ϕ) - Elem_del (ϕ)*
3. *$P,D_n \models$ PosQry*

Prova:

1. Sendo PreQry um conjunto de consultas à base de dados da TR, então pelo Lema 3.1 $P,D_0 \models PreQry$.

2. Diretamente pelo Teorema 3.4.

3. Sendo PosQry um conjunto de consultas à base de dados da TR, então pelo Lema 3.1 $P,D_n \models PreQry$. ■

A partir do lema anterior, o Teorema abaixo pode ser provado:

Teorema 3.15 (Equivalência de ações): *Sendo ϕ uma fórmula Horn-serial independente com a seguinte estrutura: $\phi \leftarrow PreQry \otimes Upd \otimes PosQry$; e α uma ação em planejamento, se pela igualdade entre conjuntos $\Delta(PreQry) = PRE$, $Elem\_ins(\phi) = IL$, $Elem\_del = DL$ e $\Delta(PosQry) = POS$, então ϕ e α são semanticamente equivalentes.*

Prova: *A prova será feita em três etapas:*

1. $P,D_0 \models PreQry \Leftrightarrow PRE \subseteq Wo$

2. $D_n = D_0 + Elem\_ins(\phi) - Elem\_del(\phi) \Leftrightarrow Wn = Wo + IL - DL$

3 $P,D_n \models PosQry \Leftrightarrow POS \subseteq Wn$

Prova para $P,D_0 \models PreQry \Leftrightarrow PRE \subseteq Wo$

1. $P,D_0 \models PreQry$	
2. $\Delta(PreQry) \subseteq O^d(D_0)$	*pelo Teorema 3.8*
3. $\Delta(PreQry) \subseteq D_0$	$O^d(D_0) = D_0$
4. $D_0 = Wo$	*pelo Teorema 3.13*
5. $PRE = \Delta(PreQry)$	*por hipótese*
6. $PRE \subseteq Wo$	*por 3, 4 e 5*

Prova para
$D_n = D_0 + Elem\_ins(\phi) - Elem\_del(\phi) \Leftrightarrow Wn = Wo + IL - DL$

1. $D_n = D_0 + Elem\_ins(\phi) - Elem\_del(\phi)$	
2. $D_0 = Wo$	*pelo Teorema 3.13*
3. $D_n = Wn$	*pelo Teorema 3.13*
4. $Elem\_ins(\phi) = IL$	*por hipótese*
5. $Elem\_del(\phi) = DL$	*por hipótese*
6. $Wn = Wo + IL - DL$	*por 1,2,3,4 e 5*

Prova para $P,D_n \models PosQry \Leftrightarrow POS \subseteq Wn$

1. $P,D_n \models PosQry$	
2. $\Delta(PosQry) \subseteq O^d(D_n)$	*pelo Teorema 3.8*
3. $\Delta(PosQry) \subseteq D_n$	$O^d(D_0) = D_0$
4. $D_n = Wn$	*pelo Teorema 3.13*
5. $POS = \Delta(PosQry)$	*por hipótese*
6. $POS \subseteq Wn$	*por 3, 4 e 5*

Mas, pela definição 3.10,
$PRE \subseteq Wo$; $Wn = Wo + IL - DL$; $POS \subseteq Wn$
É uma ação α em planejamento. ■

Um exemplo de uma fórmula independente, que descreve semanticamente uma ação, pode ser visto como um agente em um ambiente com quarto, sala e cozinha:

Pode-se definir que o agente está na sala e possui uma única ação que é mover-se de um cômodo para outro. O estado inicial D_0 pode ser descrito como:

em(sala,agente).
passagem(quarto,sala). passagem(sala,quarto).
passagem(cozinha,sala). passagem(sala,cozinha).

Observe que não há passagem entre o quarto e a cozinha. Assim, uma fórmula independente ϕ em TR descreve a ação mover_para:

Mover_para(X) ← qry(em(Y,agente)) ⊗ qry(passagem(X,Y)) ⊗ del(em(Y,agente)) ⊗ ins(em(X,agente)) ⊗ qry(em(X,agente)).

Observe que:

PreQry = qry(em(Y,agente)) ⊗ qry(passagem(X,Y))
Upd = del(em(Y,agente)) ⊗ ins(em(X,agente))
PosQry = qry(em(X,agente)).

3.3 Equivalência entre Plano (δ) em Planejamento e Fórmulas λ em TR

Definindo uma fórmula Horn-serial λ com a seguinte estrutura: $\lambda \leftarrow \phi_1 \otimes \phi_2 \otimes ... \otimes \phi_n$. Pelo Lema 3.1, se P,Do .... $D_n \models \lambda$ então, tem-se:

P,Do ... $D_1 \models \phi_1$
P,D_1 ... $D_2 \models \phi_2$
...
P,D_{n-1} ... $D_n \models \phi_n$

Para uma fórmula λ não se pode utilizar o Teorema 3.6 pelo fato da condição de independência não estar garantida. Isso ocorre pois uma fórmula ϕ pode remover ou inserir um elemento que já foi inserido ou removido por outra fórmula ϕ anterior.

Assim, a equivalência deve ser comprovada para cada fórmula ϕ presente em λ. Como descrito anteriormente, se uma fórmula ϕ é equivalente a uma ação α, então pode-se afirmar que:

Teorema 3.16 (Equivalência entre Planos) *Se para cada ϕi de λ, ϕi seja equivalente à αi, então λ é equivalente à δ.*

Prova:

Se ϕi é equivalente à αi, então para cada ϕi pode-se escrever, pelo Teorema 3.15

- $P,Do ... D_1 \models \phi_1 \Leftrightarrow PRE_1 \subseteq W_0$, $W_1 = W_0 - DL_1 + IL_1$ *e* $POS_1 \subseteq W_1$ *para* α_1
- $P,D_1 ... D_2 \models \phi_2 \Leftrightarrow PRE_2 \subseteq W_1$, $W_2 = W_1 - DL_2 + IL_2$ *e* $POS_2 \subseteq W_2$ *para* α_2
- . . .
- $P,D_n\text{-}1 ... D_n \models \phi_n \Leftrightarrow PREn \subseteq W_{n-1}$, $W_n = W_{n-1} - DL_n + IL_n$ *e* $POS_n \subseteq W_n$ *para* α_n

Mas, pela definição 3.11

- $PRE_1 \subseteq W_0$, *tem-se* $W_1 = W_0 - DL_1 + IL_1$ *e* $POS_1 \subseteq W_1$ *para* α_1
- $PRE_2 \subseteq W_1$, *tem-se* $W_2 = W_1 - DL_2 + IL_2$ *e* $POS_2 \subseteq W_2$ *para* α_2
- . . .
- $PREn \subseteq W_{n-1}$, *tem-se* $W_n = W_{n-1} - DL_n + IL_n$ *e* $POS_n \subseteq W_n$ *para* α_n

É igual a um plano δ em planejamento. ■

Um exemplo de plano modelado em TR pode ser visto considerando o exemplo da ação mover_para, onde D_0 é o mesmo do exemplo anterior com a diferença de que o agente está no quarto ao invés da sala:

em(quarto,agente).
passagem(quarto,sala). passagem(sala,quarto).
passagem(cozinha,sala). passagem(sala,cozinha).

Considere um objetivo G = {em(cozinha,agente)}, ou seja, é desejado que o agente esteja na cozinha. Assim, o seguinte plano λ satisfaz $G \subseteq Dn$, pois para:

$\lambda \leftarrow$ mover_para(sala) $\otimes$ mover_para(cozinha).

tem-se P,D0,D1,D2,D3,D4 |= λ. Onde:

D1 = D0 – {em(quarto,agente)}.
D2 = D1 + {em(sala,agente)}.
D3 = D2 – {em(sala,agente)}.
D4 = D3 + {em(cozinha,agente)}.

4 CONCLUSÃO

A Lógica de Transações vem ao encontro das necessidades dos sistemas de planejamento por possuir semântica baseada em caminhos de estado, por trabalhar com seqüência de fórmulas e por conter uma teoria que possibilita o uso de uma base de dados controlada por oráculos.

Todas essas características possibilitaram o estudo da equivalência semântica entre TR e os componentes de um sistema de planejamento. Este resultado permite definir uma base de conhecimento em Lógica de Transações, e sua conseqüente implementação em PROLOG (SANTOS, P.E., 1997), sem perda de sua estrutura semântica, como muitas vezes ocorrem com as diversas lógicas utilizadas para formalização de sistemas de planejamento.

Com a equivalência provada, torna-se possível o avanço da teoria da TR para a formalização da base de conhecimento necessária para planejamento, como feito por um estudo inicial em (TONIDANDEL E RILLO, 1998a).

Com tudo isso, pode-se concluir que a Lógica de Transações possibilita a formalização da base de conhecimento de um sistema de planejamento de modo a torná-lo independente do domínio, permitindo-se trabalhar de forma satisfatória em qualquer situação que não só as testadas previamente, e que resulte em um amplo e efetivo conhecimento das limitações do sistema, como visto em (TONIDANDEL E RILLO, 1998b) e (TONIDANDEL, 1999), e ainda permite que toda teoria desenvolvida possa ser implementada computacionalmente em linguagem PROLOG sem perda de sua semântica.

5 BIBLIOGRAFIA

BACCHUS,F. ; KABANZA,F. Planning for Temporally Extended Goals. In: AAAI-96, Portland, 1996. *Proceedings.* AAAI Press, 1996. p.1215-1222

BONNER, A.J.; KIFER, M. *Transaction logic programming*. Canadá, Toronto, University of Toronto, 1995. (Technical Report CSRI-323).

FIKES, R. E. ; NILSSON, N. J. STRIPS: A new approach to theorem proving in problem solving. *Journal for Artificial Intelligence*, v. 2, p. 189-208, 1971.

KOEHLER, J. Planning from Second Principles. *Artificial Intelligence*, v.87, 1996.

KOWALSKI, R.A.; SERGOT, M. A logic-based calculus of events. *New Generation Computing.* v. 4, p. 67-95, 1996

LIFSCHITZ, V. On the semantics of STRIPS. In: WORKSHOP REASONING ABOUT ACTIONS AND PLANS, s.l.,1987. *Proceedings*. p.1-9.

MCCARTNEY, R. Case-based planning meets the frame problem. In: AIPS-92, Maryland., 1992. *Proceedings*, Morgan Kaufmann, 1992, p.172-178

McCARTY, J.M.; HAYES, P.J. Some philosophical problems from the standpoint of artificial intelligence. 1969 In: ALLEN, J.; HENDLER, J.; TATE, A. *Readings in Planning*, San Mateo, California, Morgan Kaufmann Publishers, 1990. p. 393-495.

RILLO, M. *Uma célula robotizada com alto grau de autonomia*. 1994, p.61-105, Tese de livre docência, Escola Politécnica da Universidade de São Paulo.

SANTOS, M. V. T. On the formalization of actions using transaction logic. In: WORKSHOP ON CROSS-FERTILIZATION IN PLANNING, 12th ECAI, s.1., 1996. *Proceedings.*

SANTOS, M.V.T. ; RILLO, M. Approaching the Plans are Program paradigm using transaction logic. In: EUROPEAN CONFERENCE ON PLANNING, 4th - ECP '97, 1997. *Proceedings*. Toulouse, Sam Steel e Rachid Alami (Eds), Lecture Notes in AI: Recent Advances in AI Planning, 1997, Springer-Verlag, n.1348, p.377-389.

SANTOS, P. E. *Equivalência entre a semântica da Lógica de Transações e a semântica de sua implementação PROLOG*. São Paulo, 1997. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo.

TONIDANDEL, F. ; RILLO, M. On the treatment of the Ramification Problem with Transaction Logic. In: WORKSHOP ON INTELLIGENT MANUFACTURING SYSTEMS (IMS'98), 5th IFAC, Gramado, Brasil, 1998 (a). *Proceedings*, Elsevier editora

TONIDANDEL, F.; RILLO, M. Case-Based Planning in Transaction Logic Framework. In: WORKSHOP ON INTELLIGENT MANUFACTURING SYSTEMS (IMS'98), 5TH IFAC, Gramado, Brasil, 1998 (b). *Proceedings*, Elsevier.

TONIDANDEL, F. *Planejamento Baseado em Casos usando Lógica de Transações*. São Paulo, 1999. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo.

VERE, S. Planning in Time: Windows and durations for activities and goals. 1983. In: ALLEN, J.; HENDLER, J.; TATE, A. *Readings in Planning*, San Mateo, California, Morgan Kaufmann Publishers, 1990. p. 297-318.