

Actualização de Programas em Lógica

João Alexandre Carvalho Pinheiro Leite

Luís Moniz Pereira (orientador)

Centro de Inteligência Artificial (CENTRIA)

Departamento de Informática

Universidade Nova de Lisboa

2825 Monte da Caparica, Portugal

Sumário

A área de actualização de teorias é de importância fundamental na representação de conhecimento sobre o mundo real, caracterizado pela sua constante evolução. No que respeita à actualização dos modelos dessas teorias têm sido obtidos alguns resultados; no entanto a actualização de modelos conduz, em geral, a resultados pouco intuitivos, devido à perda de informação verificada ao substituir uma teoria pelo conjunto dos seus modelos.

Este artigo descreve a Dissertação de Mestrado “*Logic Program Updates*” [6] onde se mostra como actualizar conhecimento expresso por programas em lógica por meio de outros programas em lógica.

Os processos de actualização foram apresentados para o caso de programas normais e com negação explícita, tanto para o caso da semântica dos modelos estáveis como para a semântica bem fundada. Os resultados foram formalmente demonstrados e ilustrados com aplicações. A teoria foi implementada em Prolog.

Resumen

El area de actualización de teorias es de importancia fundamental en la representación del conocimiento sobre el mundo real, caracterizado por su constante evolución. Algunos resultados han sido obtenidos en lo que concierne a la actualización de los modelos de las teorias. No obstante los resultados son, en general, poco intuitivos en todo debido a la pérdida de información verificada al sustituir una teoria por el conjunto de sus modelos.

Este artículo describe la Disertación de Mestrado “*Logic Program Updates*” [6] donde se enseña como actualizar conocimiento expresso por programas en lógica, por medio de otros programas en lógica.

Los procesos de actualización han sido presentados en el caso de programas normales y con negociación explícita, bien para el caso de la semántica de los modelos estables así como en la semántica bien fundamentada. Los resultados están formalmente justificados e ilustrados con aplicaciones. La teoría há sido igualmente implementada.

Abstract

The field of theory update is of utmost importance in order to allow us to represent knowledge about the real world, characterised by its constant evolution. Some results have been obtained in what regards updating the models of those theories. Unfortunately, model updating leads to very unintuitive results due to the loss of information when substituting the original theory by the set of its models.

This paper describes the MSc Dissertation “*Logic Program Updates*” [6] where it is shown how to update knowledge expressed by a logic programs, by means of other logic programs.

Program updates are defined and characterised for the stable and well founded semantics, for both normal programs as well as those extended with explicit negation. All results have been formally proofed and illustrated with applications. The entire theory has been implemented.

1 Introdução

A construção de máquinas inteligentes, ou pelo menos capazes de ajudar o Homem a aumentar as suas competências racionais, é, em última instancia, o desígnio fundamental da disciplina da Inteligência Artificial (IA). Essas máquinas, mesmo operando em ambientes bastante limitados, necessitam de gerir elevadas quantidades de conhecimento. A investigação e desenvolvimento de técnicas de representação e manipulação de conhecimento desempenha assim um papel fundamental na IA. A Programação em Lógica, pela sua natureza declarativa e facilidade de implementação, tornou-se rapidamente no principal instrumento de representação de conhecimento.

Desde a década de 70, aquando da definição e implementação do *Prolog*, até aos dias de hoje, a investigação por parte da comunidade da IA tem desenvolvido a Lógica para além da fronteira do monotonicamente cumulativo, típico dos domínios fechados, precisos, completos e eternos da matemática, abrindo-a aos domínios do mundo real caracterizados por um conhecimento impreciso, incompleto, contraditório, discutível, passível de revisão, distribuído e em evolução. É precisamente em relação à capacidade de representar e manipular mundos em evolução, por oposição aos domínios eternos da matemática, que pouco foi ainda feito. Esta falta de investigação no

domínio da representação e manipulação de conhecimento dinâmico origina uma enorme lacuna aquando da tentativa de passagem dos resultados teóricos para a prática. Se a IA, através da Programação em Lógica, tem por objectivo a construção das referidas máquinas inteligentes que funcionarão, juntamente com o Homem, no mundo real caracterizado pela evolução, é indispensável investigar e desenvolver, de forma sólida e bem fundada, os mecanismos necessários à representação e manipulação de conhecimento dinâmico. Este é o tema da dissertação intitulada “*Logic Program Updates*”[6] no que toca à dinâmica da actualização de programas em lógica por outros programas em lógica.

Dada a natureza e objectivo deste artigo, bem como a limitação de espaço disponível e a extensão da dissertação abordada, optámos por dar mais ênfase à motivação e intuição em detrimento das definições e demonstrações formais, que podem ser encontradas quer ao longo da dissertação [6], quer nas três publicações que este trabalho já originou [1][7][8]. Assim, na secção seguinte apresentaremos a motivação e os objectivos que conduziram a este trabalho; na secção 3 mencionaremos os resultados obtidos, para vir concluir e elaborar sobre aplicações e desenvolvimentos futuros deste trabalho na secção 4.

2 Motivação e Objectivos

Duas formas distintas de fazer alteração de uma base de dados (BD), representada por uma teoria proposicional, foram consideradas em [5][4]. Uma delas, a *revisão*, refere-se à incorporação de conhecimento novo, acerca de um mundo estático, numa BD incompleta. A outra, a *actualização*, lida com a incorporação numa BD (incompleta ou não) de conhecimento acerca de mundos dinâmicos, i.e. em constante alteração. No presente trabalho investigamos não apenas a actualização de bases de dados mas também de bases de conhecimento (BC) representadas por programas em lógica, onde não apenas os factos (parte extensional), mas também as regras (parte intensional) variam dinamicamente. Além disso, permitimos que uma BC possa ser actualizada por outra BC, e que o processo possa ser iterado.

Alguns autores [2][9][10] abordaram o problema da actualização de programas em lógica e bases de dados, seguindo a abordagem designada por ‘*actualização de interpretações (ou modelos)*’ originalmente proposta em [4][11]. Este método reduz o problema da actualização de uma base de dados à actualização individual dos seus modelos.

Infelizmente este método de actualização sofre de várias limitações:

- Para obter a actualização BD' de uma base de dados BD , é necessário computar todos os seus modelos (tarefa bastante complicada) para

posteriormente determinar as, possivelmente múltiplas, actualizações de cada um. A actualização M_U de um dado modelo M é obtida fazendo apenas a alteração do estado dos literais em M forçada pelo programa de actualização, mantendo inalterado, por *inércia*, o estado dos restantes.

- A base de dados actualizada BD' não está directamente definida, mas sim indirectamente caracterizada como sendo uma base de dados cujos modelos são as actualizações dos modelos da BD inicial. Tal base de dados pode nem sequer existir, não havendo, em todo o caso, uma forma simples de a determinar.
- Mais importante, este método produz resultados nada intuitivos quando tanto a parte extensional (factos) como a parte intensional (regras) da base de dados são actualizadas, demonstrado pelo exemplo:

Exemplo 1 *Consideremos o seguinte programa P :*

$$P : \text{dormir} \leftarrow \neg \text{tv_on}; \quad \text{ver_tv} \leftarrow \text{tv_on}; \quad \text{tv_on} \leftarrow .$$

cujo único modelo estável é $M = \{\text{ver_tv}, \text{tv_on}\}$. Mais tarde obtemos uma actualização que nos informa que houve uma quebra de energia, e que no caso de uma quebra de energia, a televisão deixa de estar ligada, representada pelo programa de actualização:

$$U : \neg \text{tv_on} \leftarrow \text{falha_energia}; \quad \text{falha_energia} \leftarrow .$$

De acordo com a actualização de modelos, o resultado da actualização de M por U é $M_U = \{\text{falha_energia}, \text{ver_tv}\}$, ou seja, apesar de não haver energia e de a televisão não estar ligada, ainda estamos a ver televisão e não a dormir. No entanto, analisando P e U , conclui-se que M_U não representa, intuitivamente, a actualização de P por U . Dado que 'ver_tv' era verdadeiro devido a 'tv_on' ser verdadeiro, quando 'tv_on' se torna falso o mesmo deveria suceder com 'ver_tv'. Igualmente deveríamos esperar que 'dormir' se tornasse verdadeiro após a actualização. Assim, a actualização de P por U deveria ser representada pelo modelo $M'_U = \{\text{falha_energia}, \text{dormir}\}$. \diamond

Como se pode constatar no exemplo anterior, o método da actualização de interpretações não é suficiente para lidar com o caso geral em que tanto os factos como as regras são objecto de actualização. Isto deve-se à perda de informação verificada quando se substitui um programa pelo conjunto dos seus modelos. Por forma a evitar esta perda de informação, advogamos que o principio da inércia deve ser exercido sobre as regras do programa inicial,

em vez de sobre os literais dos seus modelos, no caso em que não sejam anuladas pelo programa de actualização. Voltando ao exemplo anterior, as regras "*dormir* $\leftarrow \neg tv_on$ " e "*ver_tv* $\leftarrow tv_on$ " permaneceriam válidas, por inércia, após a actualização, e seria a partir delas que se determinaria o valor dos literais "*dormir*" e "*ver_tv*".

Neste trabalho investigamos a actualização de bases de dados representadas por programas em lógica, propondo uma solução que elimina os problemas acima referidos. Isto é feito recorrendo a uma transformação que, dados um programa inicial e um programa de actualização, produz um outro programa em lógica cujos modelos obedecem a uma semântica previamente estabelecida, que caracteriza as actualizações desejadas. Esta solução contém como caso particular a actualização de modelos, no caso de uma BD puramente extensional, como seria desejável.

Posteriormente pretendemos estender os resultados obtidos de modo a iterar o processo. Isto permite a representação de um dado domínio por meio de uma sequência de programas em lógica, onde cada um representa esse domínio num dado instante. Esta noção de Programação em Lógica Iterada é, na nossa opinião, de uma importância crucial para a utilização da Programação em Lógica na modelização do mundo real, i.e. de um mundo em constante mutação.

3 Actualização de Programas em Lógica

Nesta secção e pelas razões expostas na Sec.1 apenas descrevemos intuitivamente o desenvolvimento e os principais resultados obtidos. A descrição formal e detalhada, bem como a demonstração de todos os resultados pode ser encontrada em [6]. Uma visão mais rápida pode ser obtida em [7], complementada com [8].

3.1 Caracterização Semântica

O primeiro passo na resolução do problema da actualização de um Programa em Lógica (*PL*) por meio de outro *PL* consiste em definir uma semântica, por forma a caracterizar os modelos da actualização. Como já vimos na Introdução, o processo de actualização não deve depender de um modelo particular do programa inicial (*P*), mas sim do próprio *P* (ou de um subconjunto de *P*) em conjunto com o programa de actualização (*U*).

Assim, a semântica é dada pelo conjunto de todos os modelos estáveis *M* [3] do programa $P^*(M, P, U)$ determinado pela seguinte equação de ponto

fixo:

$$P^*(M, P, U) = P_{inercial}(M, P, U) + U \quad (1)$$

$$Rejeitadas(M, P, U) = \{L \leftarrow corpo \in P : M \models corpo \\ e \exists \neg L \leftarrow corpo' \in U : M \models corpo'\} \quad (2)$$

$$P_{inercial}(M, P, U) = P - Rejeitadas(M, P, U) \quad (3)$$

onde L é um átomo A ou a sua negação $\neg A$ (onde $\neg\neg A = A$).

Ou seja, M tem que ser modelo estável de um programa constituído pelas regras de U às quais juntamos as regras de P que se mantêm válidas por inércia ($P_{inercial}$).

Esta semântica elimina o problema ilustrado pelo exemplo 1.

3.2 Transformação de Actualização

O segundo passo, com o objectivo de resolver os restantes problemas mencionados na Sec.1, consiste no desenvolvimento de uma transformação que, dados P e U , permite obter um PL tal que os seus modelos estáveis coincidam com a semântica definida em 3.1. Esta transformação (linear) define a base de dados actualizada (P_U) permitindo a determinação dos seus modelos utilizando as ferramentas disponíveis para tal.

De acordo com a transformação, P_U é composto por:

- todas as regras de U e P , após as seguintes alterações: na cabeça de todas as regras de U , substituir o literal objectivo L por um novo literal objectivo L^U ;
- na cabeça de todas as regras de P , substituir o átomo A (resp. $\neg A$) por um novo átomo A' (resp. $\neg A'$);
- regras de definição, para todo o átomo A :

$$\begin{array}{ll} A \leftarrow A', not \neg A^U; & A \leftarrow A^U; \\ \neg A \leftarrow \neg A', not A^U; & \neg A \leftarrow \neg A^U. \end{array} \quad (4)$$

As regras de definição determinam o estado de cada literal como sendo dado pela conclusão de uma regra de U , ou pela conclusão de uma regra de P , por inércia, caso não seja alterada por U . Esta transformação é completa e relevante em relação à semântica definida.

3.3 Relação com a Actualização de Interpretações

Outro resultado de grande importância é a relação estabelecida com a actualização de modelos. Assim, demonstra-se que podemos obter os resultados da actualização de uma dada interpretação M por um programa de actualização U , segundo o método da actualização de interpretações [2][9][10],

construindo um programa P que consiste apenas dos factos verdadeiros em M , actualizando-o posteriormente pelo programa U . Esta relação significa que, para bases de dados puramente extensionais, ambas as semânticas coincidem, mas que a semântica aqui apresentada generaliza P para o caso em que este representa uma BD contendo igualmente uma parte intensional.

3.4 Actualizações Iteradas

Dada uma sequência de programas em lógica $S = \{U_t : t \in T\}$, indexados por diferentes estados (t), onde cada estado pode por exemplo representar diferentes momentos do domínio descrito, o conceito de *Actualização Iterada* permite caracterizar semanticamente o estado do domínio em qualquer momento. Assim, a semântica num dado momento $t \in T$ é dada pelo conjunto de todos os modelos estáveis M do programa:

$$P_S^*(M, t) = \bigcup_{t_i \leq t} (U^{inercial}(M, t_i, t)) \quad (5)$$

onde

$$U^{inercial}(M, t_1, t) = U_{t_1} - Rejeitado(M, t_1, t) \quad (6)$$

$$Rejeitado(M, t_1, t) = \bigcup_{t_1 < t_2 \leq t} Rejeitado_i(M, t_1, t_2) \quad (7)$$

$$Rejeitado_i(M, t_1, t_2) = \{L \leftarrow corpo \in U_{t_1} : M \models corpo \\ e \exists \neg L \leftarrow corpo' \in U_{t_2} : M \models corpo'\} \quad (8)$$

ou seja, $P_S^*(M, t)$ contém as regras de todos os programas $U_{t_i}, t_i \leq t$ que se mantiveram, por inércia, até ao momento t .

Desenvolvemos igualmente uma transformação que, dada uma sequência de programas em lógica $S = \{U_t : t \in T\}$ produz um programa em lógica que pode ser consultado para determinar o estado do domínio descrito por S , em qualquer t . A transformação é completa e relevante em relação à semântica.

Este conceito permite uma grande modularização na programação em lógica, abrindo a possibilidade de integrar conhecimento obtido em diversos pontos, em alturas diferentes, dando um significado preciso à sua união.

As *Actualizações Iteradas* abrem o caminho à *Programação em Lógica Dinâmica*, rompendo com a representação de conhecimento estático da Programação em Lógica tradicional.

3.5 Outros Resultados

Entre outros, e para além dos já mencionados, este trabalho de investigação produziu os seguintes resultados:

- extensão da semântica, transformação e actualizações iteradas apresentadas para o caso da semântica bem fundada;
- extensão da semântica, transformação e actualizações iteradas apresentadas para o caso de programas em lógica estendidos com negação explícita (ou “clássica”);
- estabelecimento da actualização de programas como caso particular da actualização iterada;
- demonstração de algumas propriedades importantes da actualização de programas;
- implementação de toda a teoria desenvolvida por meio de um meta-interpretador em Prolog;
- ilustração com um exemplo realista de aplicação (para além de pequenos exemplos ao longo do trabalho).

4 Conclusões, Aplicações e Trabalho Futuro

Ao longo do trabalho de investigação, motivámos e introduzimos o conceito de actualização de programas em lógica e a sua iteração, caracterizando a semântica e definindo uma transformação de programa para os vários casos mencionados, tendo todos os resultados sido demonstrados e implementados¹.

A introdução do paradigma da Programação em Lógica Iterada (PLI) abre as portas à utilização de uma linguagem declarativa (Programação em Lógica), com todas as suas vantagens, na representação e raciocínio sobre conhecimento acerca de mundos dinâmicos, i.e. em evolução. Permite o desenho incremental e evolutivo de programas em lógica.

No que respeita a aplicações da PLI, podemos prever a sua utilização em:

- *Especificação de Software*: a utilização da Programação em Lógica como linguagem de especificação permite a representação da evolução do software, facilitando a sua actualização e manutenção;
- *Representação de Sistemas de Leis e Regulamentos*: como ilustrado em [6], a PLI é uma ferramenta ideal para representar a evolução dos sistemas de leis e regulamentos;
- *Aprendizagem*: dada a sua natureza incremental, a aprendizagem é uma das aplicações mais imediatas da PLI. De facto, podemos imaginar um sistema automático de aprendizagem de teorias por meio de um

¹O meta-interpretador que implementa as actualizações iteradas vem listado em [6] e pode ser obtido do autor (jleite@di.fct.unl.pt)

processo de geração de regras, diagnóstico sistemático e actualização da teoria;

- *Raciocínio sobre Acções*: áreas como as bases de dados activas, sistemas de produção, cálculo de eventos e cálculo de situações podem beneficiar com os conceitos aqui introduzidos, generalizando as suas abordagens;
- *Raciocínio sobre o Passado*: dado não haver qualquer perda de informação aquando do processo de actualização, contendo o programa em lógica iterado todo o historial da evolução do domínio descrito, é possível utilizá-lo para raciocinar sobre o passado, de diversas formas, como por exemplo: determinar o que poderia ter acontecido no passado com o conhecimento de que dispomos hoje; estender as regras de actualização para permitir actualizações condicionadas por estados anteriores; fazer raciocínios contrafactuais, etc.

Por último, um conjunto ordenado de programas não tem necessariamente que ser interpretado como a evolução temporal de um programa em lógica. Os diferentes módulos podem por exemplo representar diferentes conjuntos de prioridades, ou mesmo pontos de vista de diferentes agentes, abrindo assim o leque de aplicações.

Os conceitos de programação em lógica modularizada, incremental e dinâmica introduzidos neste trabalho, pelo seu potencial e flexibilidade, desempenham um papel fundamental no projecto PRAXIS de agentes racionais MENTAL, em desenvolvimento no Centro de Inteligência Artificial, Universidade Nova de Lisboa.

O mundo está em constante evolução pelo que também tem que estar qualquer forma de o representar. O conhecimento e raciocínio humanos, cuja modelização é o objectivo último da Inteligência Artificial, não só está em constante alteração, como o está de uma forma incremental. Recebemos constantemente nova informação, sobre aspectos particulares do meio em que vivemos, e automaticamente fazemos o seu processamento integrando-a no nosso conhecimento. Neste processo tentamos manter o máximo de conhecimento previamente existente, podendo no entanto parte dele ser abandonado por meio de uma reavaliação. A actualização de programas em lógica, dada a sua natureza incremental e resultados intuitivos, apresenta-se como uma boa forma de modelizar a evolução do conhecimento.

Referências

- [1] J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinski and T. C. Przymusinski. *Dynamic Logic programming*. In Sixth International Confer-

- ence on Principles of Knowledge Representation and Reasoning, KR'98, Morgan Kaufmann, 1998.
- [2] J. J. Alferes and L. M. Pereira. *Update-programs can update programs*. In J. Dix, L. M. Pereira and T. Przymusinski, editors, Selected papers from the ICLP'96 ws NMELP'96, vol. 1216 of LNAI, pages 110-131. Springer-Verlag, 1997.
 - [3] M. Gelfond and V. Lifschitz. *Logic Programs with classical negation*. In Warren and Szeredi, editors, 7th Int. Conf. on LP, pages 579-597. MIT Press, 1990.
 - [4] H. Katsuno and A. Mendelzon. *On the difference between updating a knowledge base and revising it*. In James Allen, Richard Fikes and Erik Sandewall, editors, Principles of Knowledge Representation and Reasoning: Proc. of the Second Int'l Conf., pages 230-237, Morgan Kaufmann 1991.
 - [5] A. Keller and M. Winslett Wilkins. *On the use of an extended relational model to handle changing incomplete information*. IEEE Trans. on Software Engineering, SE-11:7, pages 620-633, 1985.
 - [6] João A. Leite. *Logic Program Updates*. MSc dissertation, Universidade Nova de Lisboa, 1997.
 - [7] J. A. Leite and L. M. Pereira. *Generalizing updates: from models to programs*. In LPKR'97: ILPS'97 workshop on Logic Programming and Knowledge Representation, Port Jefferson, NY, USA, October 13-16, 1997.
 - [8] J. A. Leite and L. M. Pereira. *Iterated Logic Program Updates*. In 1998 Joint International Conference and Symposium on Logic Programming, JICSLP'98, MIT Press, 1998.
 - [9] V. Marek and M. Truszczyński. *Revision specifications by means of programs*. In C. MacNish, D. Pearce and L. M. Pereira, editors, JELIA '94, volume 838 of LNAI, pages 122-136. Springer-Verlag, 1994.
 - [10] T. Przymusinski and H. Turner. *Update by means of inference rules*. In V. Marek, A. Nerode, and M. Truszczyński, editors, LPNMR'95, volume 928 of LNAI, pages 156-174. Springer-Verlag, 1995.
 - [11] M. Winslett. *Reasoning about action using a possible models approach*. In AAAI'88, pages 89-93, 1988.