DI-PUCRJ

Série: Monografias do Departamento de Informática
MCC 24/94

Análise e Implementação da Teoria Formal de Reconhecimento de Planos

José Helano Matos Nogueira
José Jesus Perez Alcazar
Antônio Luz Furtado

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, Julho - 1994

Resumo

A parte central de um comportamento inteligente reside na capacidade do agente (homem ou máquina) elaborar planos e explicá-los. O processo de reconhecimento de planos consiste exatamente em explicar as ações observadas. A complexidade de se reconhecer planos aparece quando tentamos inferir planos razoáveis de ações que não estão explicitamente observadas, mas ações que estão implícitas no conhecimento do agente. Este processo não é puramente dedutivo, pois é usado um poderoso modelo de representação do conhecimento, heurísticas e uma biblioteca de planos típicos. O principal objetivo deste trabalho foi a implementação de um Sistema Reconhecedor de Planos (SRP), baseado na teoria formal de Kautz. Não obstante, criamos novos conceitos, regras e melhoramentos nesta teoria, criando uma teoria formal híbrida. Nosso sistema de reconhecimento de planos trabalha com informações completas e incompletas.

Abstract

The central part of intelligent reasoning resides in the agent (human or machine) capacity to realize plans and explain them. Plan recognition process consists in explain observed actions. Plan recognition complexity appears when we try to infer reasonable plan of actions which are not explicitly observed, but actions which are implicit into agent's knowledge. This process is not purely deductive because it uses a powerful knowledge representation model, heuristics, and a typical plans library. The main goal of this work was a implementation of a Plan Recognition System (PRS), based in Kautz's formal theory. Furthermore, we include new concepts, rules, and improvements in this theory, creating a hybrid theory. Our plan recognition system works with this theory treating complete and incomplete information.

1. Introdução

A parte central de um comportamento "inteligente" reside na capacidade do agente (homem ou máquina) realizar planos e explicá-los. A área de planejamento em Inteligência Artificial tem como objetivo fornecer métodos e ferramentas para tentar capturar este tipo de comportamento. O processo de planejamento pode ser dividido em duas sub-áreas: geração e reconhecimento de planos. A geração ou síntese de planos consiste em achar a solução de um problema a partir de um estado inicial, ou seja, o planejador tenta encontrar um conjunto de ações possíveis que obterá algum objetivo do plano [CoFe82, KaPe88, Nils80, Pere92 e Rich83]. Neste trabalho será abordada a sub-área de reconhecimento de planos, que de maneira geral, consiste em encontrar uma explicação das ações observadas, em termos das ocorrências delas em um ou mais planos. Porém nem todo o processo de reconhecimento consiste em explicar as ações observadas.

A complexidade do problema de reconhecimento de planos surge ao se tentar inferir planos plausíveis de ações não observadas explicitamente, mas ações que se encontram de forma implícita no conhecimento do agente. Este processo não é puramente dedutivo, pois envolve o uso de um modelo de representação do conhecimento bastante poderoso, boas heurísticas e uma biblioteca de planos típicos. Outro fator importante é que nem sempre um plano é total, ou seja, o reconhecedor ao encontrar uma explicação para uma dada ação não necessariamente finalizou seu trabalho, pois esta ação poderá fazer parte de um plano ainda maior — no caso do plano passado ao reconhecedor ser parcial. Além disso, o reconhecedor deve determinar em que ponto deve parar uma explicação, de maneira a dar ao agente somente a quantidade suficiente de informação.

Em geral, as pesquisas desenvolvidas tratam somente o que vem a ser o processo de reconhecimento de planos, deixando de mostrar os caminhos para chegar à solução deste processo. Mais recentemente, o reconhecimento de planos tem sido direcionado na parte de formalização e especificação do problema de reconhecer planos dos agentes [CSSB91, Kaut87, KoPo89, Wobc88]. Algumas destas abordagens têm se concentrado especificamente no que constitui o problema de reconhecimento de planos, sem desenvolver qualquer algoritmo. Com isso foi escolhido como ponto de partida o trabalho desenvolvido por Kautz. Em [Kaut87], pode-se encontrar o formalismo e especificação necessária, além de ser formulado um algoritmo genérico para

reconhecimento de planos. Como será visto no decorrer dos capítulos que se seguem, apesar de todo cuidado de Kautz, quanto ao seu formalismo e ao seu algoritmo, existem vários problemas que precisam e devem ser revistos em sua abordagem.

Reconhecimento de planos tem sido estudado em Inteligência Artificial através de diversos enfoques. Neste trabalho será abordado diretamente as aplicações de reconhecimento em áreas da ciência da computação, de forma que estas aplicações venham de alguma forma beneficiar o agente planejador (usuário). Uma vez que o reconhecedor encontra o plano (intensão) do usuário, este plano poderá ser utilizado na modelagem do próprio usuário, dando avisos e respondendo suas solicitações. Desta forma, o agente pode construir uma seqüência de ações independente do que o sistema inicialmente possa saber sobre elas, já que os planos são codificados em termos das relações de intensão do agente. Assim planos mal formulados poderão ser admitidos, visto que o reconhecedor encontra não somente o passo seguinte, mas também todos os possíveis planos candidatos e, caso haja alguma inconsistência, o sistema produz uma resposta cooperativa explicando os motivos da não obtenção do plano. Um ponto crítico em reconhecimento de planos é a representação das ações e objetivos que são comuns para o usuário e para o sistema, logo, em determinado momento poderão surgir diferenças entre a visão do sistema sobre os possíveis planos em um determinado contexto e a visão do usuário. Tais diferenças poderão ser contornadas através de uma interação entre o sistema e o agente.

Este trabalho tem como objetivo principal dar um enfoque teórico e prático do processo de reconhecimento de planos. A parte teórica será vista nos capítulos 2 e 3. No capítulo 2, será mostrada a forma de representar o conhecimento do agente. Portanto, serão vistos os conceitos básicos, a linguagem utilizada, a forma de tratar as diversas restrições e o grafo da hierarquia de eventos. No capítulo 3, serão abordadas as hipóteses heurísticas sobre as quais será baseada a teoria formal de reconhecimento de planos. Para isto foi utilizada a teoria formal vista em [Kaut87] e [KaPe88]. A partir desta teoria formal foram incorporados novos resultados e novos conceitos, tendo como conseqüência uma teoria de Kautz modificada. A parte prática será formada a partir desta teoria formal, visando a implementação de um reconhecedor. No capítulo 4, serão mostrados a especificação, o projeto modular. O capítulo 5 tem como objetivo mostrar a complexidade do problema e dos algoritmos propostos. Também serão mostradas as provas das hipóteses heurísticas (apêndice — A) e o código fonte do Sistema Reconhecedor de Planos—SRP implementado em linguagem PROLOG (apêndice — B).

2. Representação do Conhecimento

Existem diversas formas de se representar as ações e objetivos do agente — lógica, redes semânticas, quadros, sistema de produção e várias outras [CoFe82, Matt91, Nils80, Wins84]; algumas destas são mais expressivas e de mais fácil manipulação do que outras. Entretanto, nem todos este modelos possuem o formalismo, expressividade e simplicidade suficientes para tratar com o processo de reconhecimento de planos. Em reconhecimento de planos, as formas alternativas de representação variam muito, principalmente de aplicação para aplicação; com isso, diversos modelos foram propostos na literatura.

Em [Wobc88] o conhecimento do agente é armazenado em forma de uma hierarquia de esquema de planos, contendo as informações relevantes para captura deste conhecimento. Este esquema baseia-se em uma linguagem de cálculo de predicados, chamada STPC (Situated Typed Predicate Calculus), contendo apenas uma regra de inferência, "modus ponens", e diversos axiomas para manipulação da hierarquia de esquema de planos.

No trabalho de [LSP90], o conhecimento do agente é melhor elaborado, para isto é criado um grafo de dependência de causas, mostrando as explicações de um objetivo aceito como sendo satisfatível. De acordo com este trabalho, a base de conhecimento corresponde a um esquema de estruturas de conhecimento chamado de elementos do plano, que são hierarquicamente organizados.

O principal trabalho de reconhecimento, pode-se dizer, que foi elaborado por Kautz. Em [Kaut87] o conhecimento do agente é formado através de uma hierarquia de eventos, baseada em axiomas de fechamento não-monotonicamente declarados — Kautz define um esquema circunscritivo para reconhecimento de planos. A hierarquia de eventos é construída a partir de um grafo de explicações que é, na verdade, a estrutura responsável pelo armazenamento do conhecimento.

Todavia, a partir do modelo de Kautz, outros autores propõem modelos alternativos com intuito de ganhar maior poder, expressividade e tratabilidade do que o elaborado por Kautz.

Konolige e Pollack [KoPo89] empregam um sistema de argumentação que expande o comportamento da formalização de Kautz, assim como em [Vila90], através da retirada das hipóteses fortes. Por sua vez, em [GoCh88], o processo de reconhecimento de planos é feito através da criação de um ATMS probabilístico, chamado Wimp2. Já Goodman e Litman, [GoLi91], fazem uma mudança na estrutura de conhecimento de Kautz para que seja adaptado para o uso em interfaces inteligentes; para isto foi criado um sistema, chamado CHECS, que tem a capacidade de trabalhar com diferentes tipos de interfaces.

Conforme apresentado acima, existem diversas maneiras de representar o conhecimento do agente em um sistema de reconhecimento de planos. Em nosso trabalho foi criado um modelo híbrido a partir da teoria formal de Kautz, conforme será descrito nas seções subseqüentes.

2.1 Conceitos Básicos

Para se representar um modelo formal de reconhecimento de planos são necessários alguns conceitos concernentes à teoria a ser utilizada durante todo o trabalho.

Planos e ações são definidos como sendo *eventos*. O conhecimento do reconhecedor é representado por um conjunto de axiomas e por um conjunto de predicados da lógica de primeira-ordem, chamado de *Hierarquia de Eventos*. Esta hierarquia define as abstrações, decomposições, especializações e os relacionamentos funcionais entre os vários tipos de eventos. O relacionamento funcional, ou *papel*, inclui a relação de um evento com seus eventos *componentes.* Há um tipo especial de evento — evento *FIM*, que é formado pelos eventos que não são componentes de nenhum outro evento. O processo de reconhecer planos consiste exatamente em descrever os eventos *FIM* que foram gerados a partir de um conjunto de eventos observados.

Na abordagem utilizada por Kautz, são tratados somente as instâncias de planos nos quais os tipos de eventos aparecem na hierarquia de eventos, pois o conhecimento representado na hierarquia é completo. A priori não será tratado o processo de aquisição de novos conhecimentos, deixando este processo para um trabalho futuro. Também será mantido a completude do conhecimento, pois se a completude fosse abandonada o processo de reconhecimento de planos poderia ter um aumento incontrolável do tamanho do espaço de busca, pois um infinito número de planos poderiam ser construídos por concatenação das precondições e efeitos. Portanto, foi decidido manter a *completude de conhecimento* e somente construir planos através de abstrações, especializações e decomposições.

De acordo com a figura 1, há dois tipos de observações que levam diretamente a eventos do tipo FIM (viajar e encontrar amigo no aeroporto) e três eventos que são especializações do evento viajar (pegar ônibus, pegar trem e pegar avião), portanto são eventos que indiretamente levam a um evento FIM. Há também seis outros tipos de eventos (ir à rodoviária, embarcar, ir à estação ferroviária, comprar passagem, verificar horário e ir ao aeroporto). Na figura 1, as setas mais largas denotam uma *abstração* ou "is a" e a setas mais finas denotam um *componente* ou "part of".

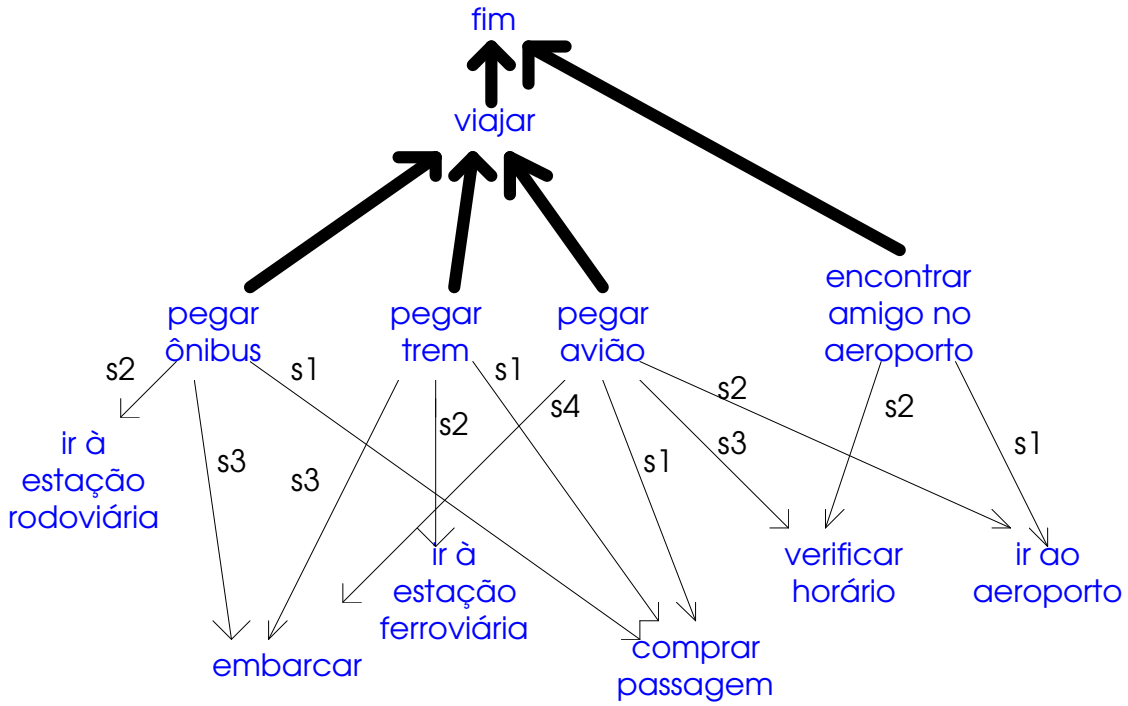

Figura 1 - Hierarquia de eventos

Os rótulos *s1, s2, s3* e *s4* servem para diferenciar os arcos componentes. Eles denominam as funções que mapeam um evento em seu respectivo componente, mostrando as partes que compõe um determinado evento. Os rótulos por si próprios não indicam formalmente a ordem temporal dos componentes, a ordem será especificada como restrições temporais que integram o axioma de decomposição.

A hierarquia de eventos da figura anterior pode ser codificada em lógica de predicados conforme abaixo:

$\forall x.\ viajar(x) \Rightarrow fim(x)$

$\forall x.\ encontrar_amigo_no_aeroporto(x) \Rightarrow fim(x)$

$\forall x.\ pegar_ônibus(x) \Rightarrow viajar(x)$

$\forall x.\ pegar_trem(x) \Rightarrow viajar(x)$

$\forall x.\ pegar_avião(x) \Rightarrow viajar(x)$

$\forall x.\ pegar_ônibus(x) \Rightarrow comprar_passagem(s1(x)) \wedge$
$ir_à_rodoviária(s2(x)) \wedge$
$embarcar(s3(x))$

$\forall x.\ pegar_trem(x) \Rightarrow comprar_passagem(s1(x)) \wedge$
$ir_à_estação_ferroviária(s2(x)) \wedge$
$embarcar(s3(x))$

$$\forall x.\ \text{pegar_avião}(x) \Rightarrow \text{comprar_passagem}(s1(x)) \wedge \text{ir_ao_aeroporto}(s2(x)) \wedge \text{verificar_horário}(s3(x)) \wedge \text{embarcar}(s4(x))$$

$$\forall x.\ \text{encontrar_amigo_no_aeroporto}(x) \Rightarrow \text{ir_ao_aeroporto}(s1(x)) \wedge \text{verificar_horário}(s2(x))$$

Os símbolos s1, s2, s3 e s4 são funções que mapeam planos ou eventos em seus respectivos componentes ou passos, como descrito anteriormente.

$$s1, s2, s3, s4 : (\text{eventos} \rightarrow \text{componentes}) \vee (\text{planos} \rightarrow \text{passos})$$

Como exemplo, suponha que uma pessoa (P) está no aeroporto, vai ao balcão de uma determinada agência de viagens e faz a seguinte pergunta: Que horas parte o avião para Fortaleza? Esta observação pode ser modelada, de acordo com os axiomas que representam o conhecimento, da seguinte forma:

$$\text{verificar_horário}(P)$$

A declaração anterior, junto com os axiomas acima, não implica que $\exists x.$ encontrar_amigo_no_aeroporto(x), ou que $\exists x.$ pegar_avião(x), ou que $\exists x.$(encontrar_amigo_no_aeroporto(x) $\vee$ pegar_avião(x)), ou ainda que $\exists x.$ fim(x); pois os axiomas permitem que se conclua somente que verificar horário é consequência de encontrar_amigo_no_aeroporto e não o contrário. Para este exemplo pode-se concluir que:

- Os eventos viajar e encontrar_amigo_no_aeroporto são especializações do evento tipo FIM, portanto, são abstraídos por FIM.

- Os axiomas possuem somente implicações ($\Rightarrow$), não possuindo bicondicionais ($\Leftrightarrow$). Logo, ao ser observado o evento verificar_horário(P) não é válido concluir que: $\exists x.$ encontrar_amigo_no_aeroporto(x), ou que $\exists x.$ pegar_avião(x), ou que $\exists x.$ (encontrar_amigo_no_aeroporto(x) $\vee$ pegar_avião(x)), pois vale somente a ida do condicional.

De acordo com a última conclusão tem-se que, se P pergunta qual o horário do vôo para Fortaleza, não quer dizer que ele pretende viajar, pois ele poderia ter ido apenas encontrar um amigo no aeroporto. Desta forma o sistema de reconhecimento deve ser inteligente ao ponto de fazer uma interação com o usuário, com o intuito de dar somente as informações necessárias ao agente que gerou o plano. Assim, o sistema de reconhecimento deve perguntar se o agente (no caso, P) vai viajar, se este for viajar então o reconhecedor fornece todas as informações necessárias a sua viagem: verificação do horário do vôo, o nome do dono da passagem, o número da passagem e número do portão de embarque. Caso contrário, o sistema informa somente o horário do vôo e o local de partida do mesmo.

Agora suponha que o reconhecedor observa que ocorrem os eventos: {verificar_horário(merlim), ir_ao aeroporto(arthur)}, onde "merlim" e "arthur" são constantes. Agora é dado a crença ao sistema de que:

$$\forall x.(\text{encontrar_amigo_no_aeroporto}(x)) \equiv \text{ir_ao_aeroporto}(s1(x)) \wedge \text{verificar_horário}(s2(x)).$$

A crença que $\exists x.\text{encontrar_amigo_no_aeroporto}(x)$ ainda não é válida, pois falta a premissa que diz que $\exists x.(\text{merlim}=s2(x) \wedge \text{arthur}=s1(x))$.

É possível fortalecer a crença acima, afirmando-se que, sempre que alguém vai ao aeroporto e verifica o horário ela está querendo encontrar um amigo, formalmente:

$$[\exists x.\text{encontrar_amigo_no_aeroporto}(x)] \equiv [\exists y,z.(\text{ir_ao_aeroporto}(y) \wedge \text{verifica_horário}(z))]$$

Esta mudança permite reconhecer a conclusão que $\exists x.$ encontrar_amigo_no_aeroporto(x), todavia não soluciona todos os problemas. Primeiro, nem sempre é possível dar condições suficientes para todo tipo de evento — o número de eventos são infinitos. Segundo, o reconhecedor ainda necessitará observar todos os passos do plano com as modificações que foram incluídas, o que gera um complexidade combinatorial. Terceiro, não é possível que a todo momento o sistema tenha que receber hipóteses que diminua o seu universo de discurso sobre o problema, tornar-se-ia inviável para aplicações mais realistas, o que contraria a motivação inicial de reconhecimento de planos. Por último, neste exemplo, com o último fortalecimento do axioma que representa a crença, não seria possível que uma pessoa se dirigisse ao aeroporto, verificasse o horário do vôo e fosse viajar.

A geração de hipóteses completas, para uma dada hierarquia, podem ser obtidas apenas com a introdução de alguns novos axiomas. No exemplo anterior, a hipótese necessária é

$$\forall x.\text{verifica_horário}(x) \rightarrow (\exists y.\text{pegar_avião}(y) \wedge x = s3(y)) \vee \exists y.\text{encontrar_amigo_no_aeroporto}(y) \wedge x = s2(y))$$

A completude das hipóteses acima é equivalente a fazer *circunscrição* da hierarquia de eventos, conforme [McCa80]. Kautz também utiliza-se de uma teoria de modelos de circunscrição para provar a teoria dos modelos de reconhecimento de planos, para isto ele cria conceito de *modelos cobertos*. Modelos cobertos vem do fato de que todo evento em cada modelo é explicado ou "coberto" por algum evento FIM que o contém. No exemplo anterior, os únicos modelos cobertos são isomórficos a um dos seguintes modelos:

{fim(c),viajar(c),pegar_avião(c), comprar_passagem(s1(c)), ir_ao_aeroporto (s2(c)), verificar_horário(s3(c)),embarcar(s4(c))} ou

{fim(c),encontrar_amigo_no_aeroporto(c), ir_ao_aeroporto(s1(c)), verificar_horário(s2(c))}

Qualquer modelo contendo apenas uma instância de verifica_horário, mas que não corresponde a um evento FIM não é um modelo coberto.

Quando diversos eventos são observados, algumas hipóteses precisam ser acrescentadas. Suponha que os eventos

{verificar_horário(merlim), comprar_passagem(olimpia)}

foram observados. Estas duas observações junto com as hipóteses

$$\forall x.\text{verifica_horário}(x) \rightarrow (\exists y.\text{pegar_avião}(y) \wedge x = s3(y)) \vee \exists y.\text{encontrar_amigo_no_aeroporto}(y) \wedge x = s2(y))$$

$$\forall x.\text{comprar_passagem}(x) \rightarrow (\exists y.\text{pegar_ônibus}(y) \wedge x = s1(y)) \vee \exists y.\text{pegar_trem}(y) \wedge x = s1(y)) \vee \exists y.\text{pegar_avião}(y) \wedge x = s1(y))$$

não implicam que uma instância de pegar_avião ocorre. A primeira observação poderia ser um passo para pegar_avião ou encontrar_amigo_no_aeroporto. A segunda poderia ser um passo de um plano para pegar_ônibus, ou pegar_trem, ou pegar_avião. Apesar disso, é razoável assumir que todas as observações partem do mesmo evento FIM. Dada a hipótese que

$$\forall x,y.\ \text{fim}(x) \wedge \text{fim}(y) \rightarrow x = y$$

a conclusão que $\exists x.\text{pegar_avião}(x)$ é dedutivamente verdadeira, que por sua vez significa que $\exists x.\text{viajar}$, já que o evento pagar_avião é abstraído por viajar. Em teoria dos modelos, esta última hipótese corresponde a selecionar dentre os modelos cobertos aqueles que contém um número mínimo de eventos FIM. Estes modelos são chamados de *modelos cobertos mínimos* das observações e da hierarquia.

2.2 Linguagem

A linguagem utilizada foi basicamente a lógica de primeira ordem com a igualdade. Entretanto, para provar as hipóteses básicas da teoria formal para reconhecimento de planos, torna-se necessário um conhecimento de lógica não-monotônica, no caso, *Cinscunscrição*. Para isto, ver [Gins87, Luka90, McCa80, Reit87].

Primeiro foi introduzida uma semântica padrão para esta linguagem e em seguida a linguagem foi estendida para tratar com o problema de reconhecimento de planos. Logo, alguns conceitos básicos são mostrados a seguir para que se possa obter um melhor entendimento da linguagem utilizada.

Modelo : Interpreta a linguagem e mapeia termos em indivíduos, funções em tuplas de indivíduos para indivíduos e predicados em um conjunto de tuplas de indivíduos. Isto é, se M é um modelo, então este mapeamento é feito aplicando-se M a um termo, função ou predicado.

Domínio : Domínio de discurso de um modelo M, **dom(M)**, é o universo sobre o qual os termos, variáveis, constantes e funções ou predicados são válidos. É através da teoria dos modelos que se captura o sentido semântico das assertivas de primeira-ordem.

Por exemplo, para algum modelo M:

gosta(estudar(nectanebo), magias) é verdadeiro $\Leftrightarrow$
M[nectanebo] $\in$ M[estudar],
M[(estudar(nectanebo), magias) $\in$ M[gosta]

Logo, caso isto ocorra, então M[nectanebo] $\in$ dom(M).

Meta-variáveis : São variáveis extra-linguagem. Estão diferenciadas nesta teoria formal de reconhecimento de planos por "**:**" colocado antes do nome da variável. Por exemplo, :A, :H, :C, ...

Instanciação : É a atribuição de constantes para variáveis do modelo. A expressão M {x/:C} significa que o modelo é semelhante a M, exceto que a variável x é mapeada para o indivíduo :C.

Na parte não monotônica da linguagem, foram abordados apenas alguns conceitos e sintaxe básica da lógica de Circunscrição. Uma forma de realizar a não monotonicidade é tratar como teoremas aquelas sentenças que são verdadeiras em todos os modelos da teoria. Porém, ampliando-se a teoria podem aparecer novos modelos distintos dos já existentes, então o que uma vez era teorema pode não ser mais; pois alguns teoremas existentes anteriormente podem ser invalidados pelos novos modelos. A Circunscrição utiliza essa forma de tratar com modelos, portanto requer que os modelos satisfaçam à propriedade de minimalidade [McCa80, McCa86]. A relação entre Circunscrição e reconhecimento de planos será feita de maneira precisa, através do desenvolvimento de uma teoria própria.

Em reconhecimento de planos, Circunscrição é uma transformação sintática de um conjunto de sentenças representando o conhecimento do agente. Seja $S[\pi]$ um conjunto de fórmulas contento uma lista de predicados π. A expressão $S[\delta]$ é o conjunto de fórmulas obtido pela re-escrita de S com cada membro de π trocado pelo membro correspodente em δ. A expressão $\delta \leq \pi$ declara que a fórmula que é a extensão de cada predicado em δ é um subconjunto da extensão do predicado correspondente em π , isto é,

$$(\forall x.\delta_1(x) \rightarrow \pi_1(x) \wedge \ldots \wedge \forall x.\delta_n(x) \rightarrow \pi_n(x)),$$

onde x é uma lista de variáveis que serve como argumento para cada δ_i. A Circunscrição de π relativo a S é escrita por **circun(S,π)**. Suponha que M1 e M2 são modelos de S e que são idênticos, exceto que a extensão em M2 de um ou mais predicados em π é um subconjunto próprio das extensões daqueles predicados em M1. Isto é denotado por **M1 >>M2**. Assim, diz-se que M1 é ***mínimo*** em π relativo a S se não existe tal M2. A circunscrição circun(S,π) é verdadeira em todos os modelos de S que são mínimos em π. A completude da Circunscrição será assumida como sendo verdadeira, pois foge ao escopo deste trabalho.

Os conectivos proposicionais e os predicados são interpretados da maneira usual da lógica. Eventualmente será referenciado provas que exigirão um conhecimento do método de *dedução natural*. Também será utilizado um conjunto de símbolos constantes, chamadas *constantes de Skolem*, que serão usadas na regra do quantificador existencial. Estas constantes, por convenção, serão diferenciadas por um * , colocado antes. Nenhuma constante de Skolem deverá aparecer no final da prova. Para um melhor embasamento desta parte ver [Ende73, Dale89]. Entretanto, nenhum conhecimento profundo será necessário.

2.3 Representação do Tempo

Uma parte importante em reconhecimento de planos é a inclusão de restrições temporais no modelo de representação da hierarquia de eventos. Isto é um fator crucial no reconhecimento de uma observação, visto que todo evento possui uma relação com seu tempo de ocorrência e, muitas vezes, planos candidatos — planos consistentes com uma possível solução, podem ser eliminados apenas com uma verificação em suas restrições temporais. Uma análise temporal, usando algum formalismo, deve permitir o reconhecimento de instâncias que estão registradas nos relacionamentos existentes na estrutura de representação do conhecimento. Entretanto, o agente no qual o conhecimento foi modelado pode fornecer informações a respeito das restrições de tempo entre as suas ações. Estas restrições podem diferir das restrições que se encontram armazenadas no sistema, logo o sistema de reconhecimento precisa de técnicas para tratar, de maneira correta e coerente, tais restrições, de modo a manter a sua base de dados consistente.

Muitos trabalhos desenvolvidos têm esquecido a representação do tempo em suas abordagens. Dentre os paradigmas que tratam o problema temporal vale a pena destacar os que dão um tratamento adequado a este problema e que contribuiram de alguma forma com este trabalho.

Konolige e Pollack em [KoPo89] tratam o aspecto temporal utilizando uma simplificação da estrutura temporal através da adoção do cálculo de situações de ações atômicas. Para isto, as proposições são consideradas como sendo verdadeiras, dependendo da situação.

Na teoria lógica de reconhecimento de planos de Wobcke, vista em [Wobc88], a descrição de tempo também utiliza um cálculo de situações, no caso o STPC (Situated Typed Pedicate Calculus). Pelo ponto de vista da implementação, é criado um "frame" onde alguns dos componentes são as restrições temporais; porém o próprio STPC falha em seu formalismo, não abordando como é feito a definição e manipulação destes axiomas.

No uso em assistentes inteligentes, [LSP90], foi assumido como convenção que quando os argumentos de uma cláusula são eventos que incluem tempo, então o primeiro evento que aparece precede o segundo e assim por diante. Outra convenção é que os eventos ocorrem em pontos de tempo (instantes), eles não têm duração, em contrário aos intervalos que possuem duração temporal.

Kautz, em sua abordagem tradicional, encontrada em [Kaut87], utiliza-se da álgebra de intervalos de Allen, ver [Alle83]. De acordo com este formalismo tem-se que o tempo é linear, que os intervalos de tempo são individuais e que podem ser identificados como pares de números racionais usados em diversas relações, conforme pode ser visto na figura 2:

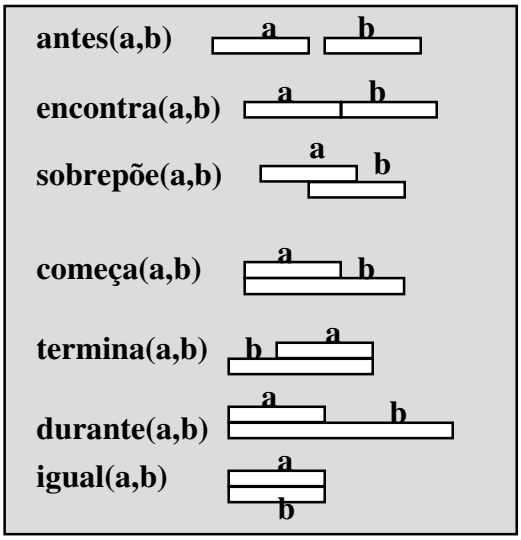

Figura 2 - Relações Temporais

Assim como em Kautz, [CSSB91], utilizam o cálculo de intervalos de Allen, todavia Cohen et al classificam as primitivas de Allen em duas relações de mais alto nível:

precede(a,b) = antes(a,b) ∨ encontra(a,b) ∨ sobrepõe(a,b)
inclui(a,b) = começa(b,a) ∨ durante(b,a) ∨ termina(b,a) ∨ igual(a,b)

Além destas relações, é criado uma grafo de recobrimento de tais ralações.

Estas duas relações ajudam a organizar a ordem temporal da hierarquia de eventos como uma representação em grafo, onde as ligações (linhas) cheias são as

relações "inclui" (i) e as ligações tracejadas (p) são as relações "precede", conforme pode ser visto na figura 3.

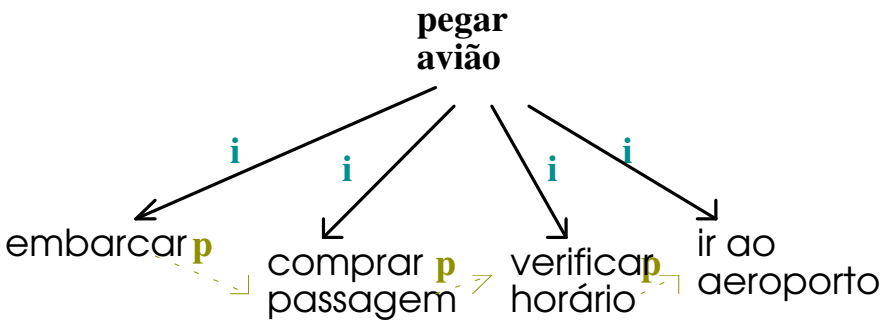

Figura 3 - Relações inclui e precede

Nesta forma de representação é possível concluir que a relação de decomposição pode ser descrita como sendo a relação "inclui", desde que o tempo de execução para uma ação abstrata inclua o tempo para seus passos de mais baixo nível. Ademais, para a maioria dos planos as relações temporais entre os passos de uma abstração podem ser registrados como "precede".

2.4 Hierarquia de Eventos

Uma Hierarquia de Eventos é uma coleção de axiomas que podem ser vistos como uma codificação lógica de uma rede semântica. Estes axiomas representam a relação de abstração e decomposição entre os tipos de eventos.

Pode-se definir uma hierarquia de eventos H como sendo uma quíntupla :

$H = <H_E, H_A, H_{EB}, H_D, H_G>$

- H_E é o conjunto de predicados unários de um tipo de evento. H_E contém diferentes predicados do tipo FIM e TOTAL. Todo membro da extensão de um predicado é chamado evento "produção". Evento TOTAL, é o tipo de evento mais geral e FIM é o tipo de todos os eventos que não são parte de nenhum evento mais abrangente.

- H_A é o conjunto de axiomas de abstração da forma :

  abs(E_j,E_i) ou **is_a(E_i, E_j)** ou **E_j abstrai E_i**

Formalmente,

Sejam dois eventos quaisquer E_i, $E_j \in H_E$. Têm-se que E_i abstrai E_j , se e somente se, para todo evento E_i, temos E_j, mas não vale o contrário. Valendo também a transitividade em lógica de segunda-ordem. Ou seja, diz-se que E_i abstrai E_i, se e somente se, são satisfeitos os seguintes axiomas:

(I) $\forall E_i, E_j\ [abs(E_j, E_i) \leftrightarrow (\forall x.E_i(x) \rightarrow E_j(x))]$

(II) $\forall E_i, E_j, E_k\ [abs(E_i, E_j) \wedge abs(E_j, E_k) \rightarrow abs(E_i, E_k)]$

Caso seja necessário que os eventos dos axiomas acima sejam do mesmo tipo, então basta modificar o axioma (I) da seguinte forma (usando lógica poli-sortida):

(I) $\forall E_i, E_j\ [abs(E_j, E_i) \leftrightarrow ((\forall x:t).E_i(x) \rightarrow E_j(x))]$, $t \in$ conjunto de tipos

Note que a redução deste axioma para lógica de primeira-ordem poderá ser realizada usando relativização:

$[(\forall x:t)\ \varphi]^{*1} := \forall x[tipo_t(x) \rightarrow \varphi]$

O simétrico da abstração é a especialização, que pode ser definida como sendo:

esp(E_i , E_j), ou E_i especializa E_j

Formalmente, para dois eventos quaisquer, E_i, $E_j \in H_E$. Têm-se que E_i especializa E_j , se e somente se, são satisfeitos os seguintes axiomas:

(I) $\forall E_i, E_j\ [esp(E_i, E_j) \leftrightarrow (\forall x.E_i(x) \rightarrow E_j(x))]$

(II) $\forall E_i, E_j, E_k\ [esp(E_i, E_j) \wedge esp(E_j, E_k) \rightarrow esp(E_i, E_k)]$

Caso seja necessário, assim como na abstração, que os eventos dos axiomas acima sejam do mesmo tipo, então basta modificar o axioma (I) da seguinte forma:

(I) $\forall E_i, E_j\ [abs(E_j, E_i) \leftrightarrow ((\forall x:t)^* .E_i(x) \rightarrow E_j(x))]$, $t \in$ conjunto de tipos

Uma observação importante que vale a pena destacar é:

Se $\forall x.E_i(x) \rightarrow E_j(x)$
Então $esp(E_i, E_j) \wedge$
$abs(E_j, E_i) \wedge$
$\neg abs(E_i, E_j)$

---

[1] * Significa a relativização do axioma em lógica poli-sorti
para lógica de primeira-ordem.

Graficamente,

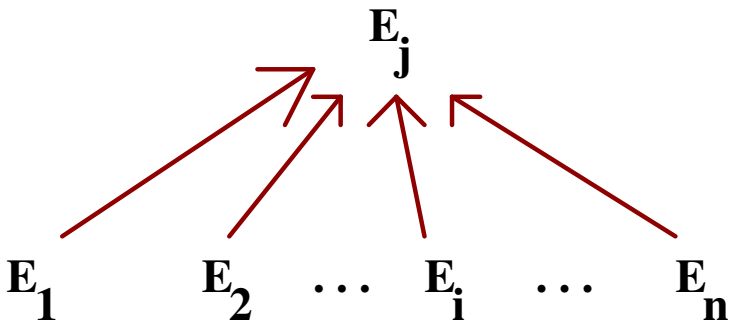

Figura 4 - Relações de abstração e especialização

♦ H_{EB} é o conjunto de eventos do tipo base, isto é, aqueles membros de H_E que não abstrai nenhum outro evento. São os eventos que aparecem na base da hierarquia de eventos.

$$H_{EB} = H_E — H_A$$

♦ H_D é o conjunto de axiomas de decomposição da forma :

$$(\forall x.E_0(x) \rightarrow E_1(f_1(x)) \wedge \ldots \wedge E_n(f_n(x)) \wedge \kappa \,,$$

onde

$E_1,\ldots,E_n \in H_E$. ; $f_1, \ldots, f_n$ são regras funcionais ou passos e κ é uma subfórmula contendo elementos que não sejam membros de H_E.

A fórmula κ descreve as *restrições* sobre E_0. De E_1 até E_n são eventos chamados *componentes diretos* de E_0. Portanto, nenhum evento FIM ou qualquer especialização dele pode aparecer como um componente direto.

♦ H_G é o conjunto de axiomas gerais, ou seja, aqueles axiomas que não contêm nenhum membro de H_E. H_G inclui os axiomas de restrições temporais, assim como qualquer outro fato que não foi especificamente declarado nos eventos.

Algumas considerações importantes devem ser destacadas na hierarquia de eventos:

Eventos *compatíveis* - Dois tipos eventos são ditos compatíveis se há um evento que ambos abstraem ou são iguais a ele.

Graficamente:

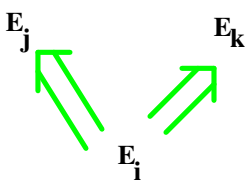

Figura 5 - E_j é compatível com E_k

Parâmetros - Os parâmetros de um evento produção são os valores das regras funcionais (atributos) mencionadas nos axiomas de decomposição.

Componente direto - Um evento produção é um componente direto de um outro somente no caso em que ele é igual ao valor de alguma das regras funcionais aplicáveis à produção formadora (origem).

Graficamente:

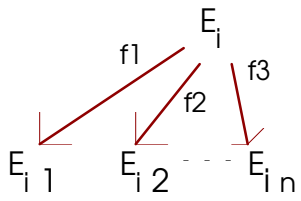

Figura 6 - Componentes diretos

Onde, E_{i1}, E_{i2}, . . . , E_{in} são componentes diretos de E_i e f_1, . . . , f_n são os passos.

Formalmente:

Suponha que :Ci e :Co são eventos produção. Diz-se que :Ci é um componente direto de :Co em M, se e somemte se ,

1. $\exists(Ei,Eo)$ / :Ci $\in$ M[Ei] $\wedge$:Co $\in$ M[Eo],

2. H_D contém um axioma da forma

$$\forall x.Eo(x) \rightarrow E_1(f_1(x)) \wedge \ldots \wedge E_i(f_i(x)) \wedge \ldots \wedge E_n(f_n(x)) \wedge \kappa ,$$

3. :Ci = M[f_i](:Co)

A Relação de *componente* é o fechamento transitivo da relação de componente direto. O fato que um :Cj é o mesmo ou um componente de :Co é descrito por :Cj **componente =** :Co.

Exemplo de uma Hierarquia de eventos (O mundo da Arte Culinária)

O exemplo da figura 6 é uma modificação do exemplo encontrado em [Kaut87]. As ações envolvidas no mundo da arte culinária formam um domínio interessante para o reconhecimento de planos. As relações de especialização entre os vários tipos de comida são refletidas pelas relações de especialização entre as ações que criam uma determinada refeição. As decomposições são associadas com as ações de preparar um tipo de comida de maneira que mostre os passos na preparação da comida. Um bom cozinheiro

armazena as informações nos vários níveis da hierarquia de abstração que ele possui. Por exemplo, o cozinheiro sabe que certas ações precisam ser efetuadas para fazer um determinado tipo de molho, assim como certas restrições deverão ser satisfeitas durante a preparação. Uma especialização de um tipo de molho adiciona passos e precondições, por exemplo, em um determinado momento da receita deve ser adicionada uma determinada quantidade de extrato de tomate.

Será assumido que o cozinheiro e o observador possuem o mesmo conhecimento a respeito de culinária — por exemplo, ambos conhecem um livro de receitas hierarquicamente arrumado. As ações do cozinheiro serão anotadas pelo observador que, tenta inferir o que o cozinheiro está fazendo. Um cozinheiro pode preparar diferentes pratos ao mesmo tempo, logo não será possível assumir que todas as observações são parte da mesma receita. Diferentes eventos FIM podem compartilhar alguns passos na preparação de um prato.

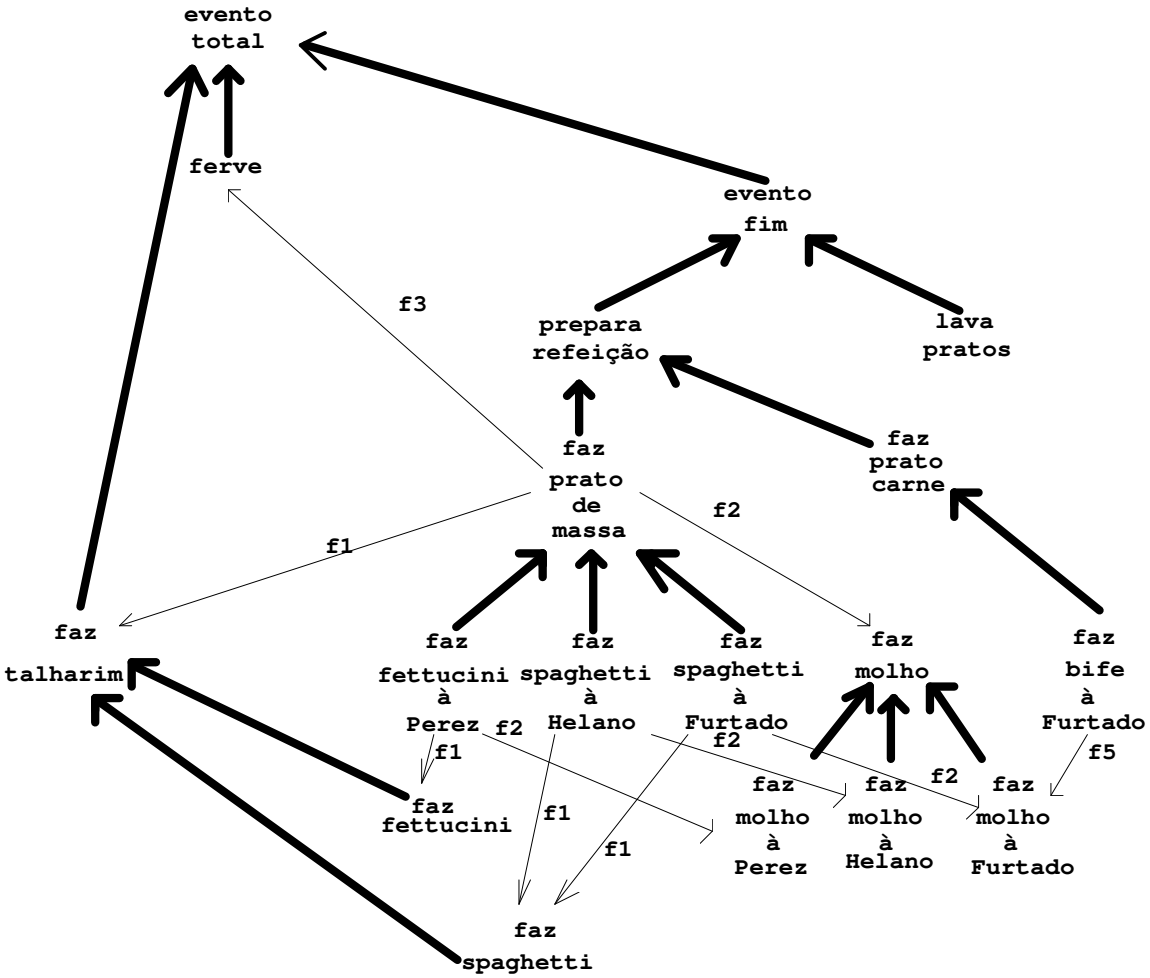

Figura - 6 Hierarquia de Eventos para o domínio de arte culinária

A figura 6 ilustra uma hierarquia típica de arte culinária. As *setas largas* representam abstrações e as *setas estreitas* representam os componentes. Ambas são meta-relações. É importante notar que a hierarquia de abstrações é codificada pelos axiomas H_A e a hierarquia de decomposições pelos axiomas H_D, porém apesar de HA e

H_D estarem relacionados eles são axiomas separados. A figura 6 ilustra apenas algumas das informações do domínio de arte culinária.

A Descrição formal da hierarquia de eventos da figura 6 pode ser vista a serguir :

Seja Faz = {faz_talharim, faz_fettucini, faz_spaghetti,
faz_prato_de_massa,faz_fettucini_à_Perez, faz_spaghetti_à_Helano,
faz_spaghetti_à_Furtado, faz_prato_de_carne, faz_bife_à_Furtado}

- H_E = Faz $\cup$ evento_fim $\cup$ {prepara_refeição, lava_pratos, ferve}

- H_A relaciona os tipos de eventos com suas especializações. Este relaciomamento é feito declarando-se que toda instância do tipo mais especializado é uma instância do tipo mais abstrato. Logo H_A será composto pelos axiomas:

H_A = { $\forall$x. faz_talharim(x) $\rightarrow$ evento_total(x),
$\forall$x. ferve(x) $\rightarrow$ evento_total(x),
$\forall$x. evento_fim(x) $\rightarrow$ evento_total(x),
$\forall$x. faz_fettucini(x) $\rightarrow$ faz_talharim(x),
$\forall$x. faz_spaghetti(x) $\rightarrow$ faz_talharim(x),
$\forall$x. prepara_refeição(x) $\rightarrow$ evento_fim(x),
$\forall$x. lava_pratos(x) $\rightarrow$ evento_fim(x),
$\forall$x. faz_prato_de_massa(x) $\rightarrow$ prepara_refeição(x),
$\forall$x.faz_prato_de_carne(x) $\rightarrow$ prepara_refeição(x),
$\forall$x.faz_fettuticini_à_Perez(x) $\rightarrow$ faz_prato_de_massa(x),
$\forall$x.faz_spaghetti_à_Helano(x) $\rightarrow$ faz_prato_de_massa(x),
$\forall$x.faz_spaghetti_à_Furtado(x) $\rightarrow$ faz_prato_de_massa(x),
$\forall$x.faz_bife_à_Furtado(x) $\rightarrow$ faz_prato_de_carne(x),
$\forall$x.faz_molho_à_Perez(x) $\rightarrow$ faz_molho(x),
$\forall$x.faz_molho_à_Helano(x) $\rightarrow$ faz_molho(x),
$\forall$x.faz_molho_à_Furtado(x) $\rightarrow$ faz_molho(x)}

Note que a relação de transitividade também é valida

- H_{EB} é formado por todos os eventos que aparecem na base da hierarquia de abstração.

- H_D é o conjunto de axiomas da forma:

$(\forall x.E_0(x) \rightarrow E_1(f_1(x)) \wedge \ldots \wedge E_n(f_n(x)) \wedge \kappa.$

No exemplo da figura 6, levando-se em consideração $H_D - \kappa$, tem-se que

H_D = {$\forall$x.faz_prato_de_massa(x) $\rightarrow$ faz_talharim(f1(x)) $\wedge$
faz_molho(f2(x)) $\wedge$
ferve(f3(x)) $\wedge$

$\forall x.faz_fettucini_à_Perez(x) \rightarrow faz_fettucini(f1(x)) \wedge$
$faz_molho_à_Perez(f2(x))$
$\wedge$
$\forall x.faz_spaghetti_à_Helano(x) \rightarrow faz_spaghetti(f1(x)) \wedge$
$faz_molho_à_Helano(f2(x))$
$\wedge$
$\forall x.faz_spaghetti_à_Furtado(x) \rightarrow faz_spaghetti(f1(x))$
$faz_molho_à_Furtado(f2(x))$
$\wedge$
$\forall x.faz_bife_à_Furtado(x) \rightarrow faz_molho_à_Furtado(f5(x))$

Note que não foi enumerado κ, pois compreende as restrições, precondições e efeitos sobre os eventos que não aparecem em H_E. Assim κ depende somente da forma como são organizadas as informações pelo agente. Por exemplo, o axioma de decomposição completo para o evento fazer um prato de massa pode ser da seguinte forma:

Componentes :
$\forall x.faz_prato_de_massa(x) \rightarrow faz_talharim(f1(x)) \wedge$
$faz_molho(f2(x)) \wedge$
$ferve(f3(x)) \wedge$

Restrições de igualdade :
$agente(f1(x)) = agente(x) \wedge$
$resultado(f1(x)) = entrada(f3(x)) \wedge$

Restrições de tempo :
$durante(tempo(f1(x)), tempo(x)) \wedge$
$antes(tempo(f1(x)), tempo(f3(x))) \wedge$
$depois(tempo(x),pos_tempo(x)) \wedge$

Precondições :
$na_cozinha(agente(x),tempo(x)) \wedge$
$sabe_fazer_massa(agente(x)) \wedge$

Efeitos :
$prato_de_massa(resultado(x)) \wedge$
$pronto_massa(resultado(x),pos_tempo(x))$

Os nomes dos componentes funcionais f1, f2, . . . , fn são arbitrários e não têm qualquer ordem temporal. Os eventos que especializam um evento Ei adicionam novas restrições, funções e precondições em Ei.

- H_G são os eventos gerais e podem ser obtidos pela fórmula abaixo:

$H_G = \{ \forall Ei / Ei \notin H_E \Rightarrow Ei \in H_G \}$

3. Hipóteses Heurísticas de Kautz

Foi visto que os tipos de inferências realizadas no processo de reconhecimento de planos não são obtidas somente da hierarquia de eventos. A hierarquia de eventos é direcionada para assumir que não existem eventos fora de H_E e que todas as relações de abstração estão em H_A. As decomposições por sua vez são direcionadas para assumir que todos os eventos do tipo não "fim" ocorrem somente como componentes de outros eventos. Estas hipóteses são aceitáveis, visto que é assumido que a hierarquia codifica todo o conhecimento sobre os eventos. Se a hierarquia é ampliada, as hipóteses devem ser revisadas. Foram usadas quatro hipóteses heurísticas (EXA, DJA, CUA e MCA) com o intuito de combinar as informações das diversas observações, de maneira que seja possível inferir sobre o conhecimento implícito do agente representado na hierarquia.

3.1 Exaustividade (EXA)

O conjunto EXA de todas as hipóteses de exaustividade é composto de todas as declarações da forma :

$$\forall x.E_0(x) \rightarrow (E_1(x) \vee \ldots \vee E_n(x))$$

onde

Eo é um predicado de H_A e

{E1, . . . , En} são todos os predicados diretamente abstraídos por Eo.

Esta hipótese permite determinar a ocorrência de um evento por eliminação de todas as outras possibilidades. Na hierarquia da figura 6, suponha que o agente está fazendo algum tipo de molho que não é o molho à moda Perez, nem o molho à moda Helano. Logo conclui-se que o agente esta fazendo molho à moda Furtado. Esta conclusão é justificada por EXA. No caso, a hipótese é que

$$\begin{aligned}\forall x.\text{faz_molho}(x) \rightarrow\ & \text{faz_molho}(x)\text{_à_Perez}(x) \vee \\ & \text{faz_molho}(x)\text{_à_Helano}(x) \vee \\ & \text{faz_molho}(x)\text{_à_Furtado}(x).\end{aligned}$$

É possível demonstrar que EXA é equivalente a uma circunscrição da hierarquia de eventos, veja apêndice — A.

3.2 Disjunção (DJA)

O conjunto de hipóteses DJA consiste de todas as declarações da seguinte forma:

$\forall x.(\sim E_1(x) \vee \sim E_2(x))$, onde E_1 e E_2 são eventos **não** compatíveis.

Na figura 6, suponha que o agente está fazendo um prato de massa. Portanto, não se pode concluir que este evento é uma instância de fazer um prato de carne. Isto é, foi assumido a hipótese que

$\forall x.$ faz_prato_de_massa(x) $\rightarrow \sim$faz_prato_de_carne(x)

Suponha agora que foi adicionado que um novo tipo na hierarquia (faz_raviolli) tal que especializa faz_prato_de_massa :

$\forall x.$ faz_raviolli(x) $\rightarrow$ faz_prato_de_massa(x)
$\forall x.$ faz_raviolli(x) $\rightarrow$ faz_prato_de_carne(x)

Portanto, a hipótese que prato de carne e prato de massa são disjuntas não é mais válida. Entretanto, assumindo-se que o conhecimento sobre os evento é completo então é razoável concluir que dois tipos de eventos são disjuntos, a menos que um abstraia o outro ou que ambos abstraiam um tipo de evento comum (são compatíveis).

A hipótese de disjunção junto com a hipótese de exaustão resulta que todo evento tem um único tipo básico, onde os tipos base são as folhas da hierarquia de abstração. Assim como em EXA, também é possível demonstrar DJA usando circunscrição de eventos da hierarquia de eventos. Veja apêndice — A.

3.3 Componente / Uso (CUA)

Uma das hipóteses mais importante para o processo de reconhecimento de planos é a CUA. Ela permite que seja deduzido a disjunção das possíveis causas para a ocorrência de um evento. Esta hipótese declara que um plano ou ação implica na disjunção dos planos que a usam como componente. A definição do conjunto CUA das hipóteses de componente/uso é formulada por :

i. Defini-se **compatível(E)** como o conjunto dos eventos em que E é compatível, $\forall E \in H_E$

ii. Considere todos os axiomas de decomposição em que todo membro de compatível(E) aparece do lado direito deste axioma. O j-ésimo axioma de decomposição tem a seguinte forma :

$\forall x.E_{j0}(x) \rightarrow E_{j1}(f_1(x)) \wedge \ldots \wedge E_{ji}(fj_i(x)) \wedge \ldots \wedge E_{jn}(f_{jn}(x)) \wedge \kappa$

onde E_{ji} é membro de compatível(E)

iii. Suponha que vários destes axiomas, definidos no item ii, são de comprimento m, para m > 0, então a fórmula CUA para o evento E é dada por:

$$\forall x.E(x) \rightarrow fim(x) \vee$$
$$(\exists y.E_{1,0}(y) \wedge f_{1i}(y) = x) \vee$$
$$(\exists y.E_{2,0}(y) \wedge f_{2i}(y) = x) \vee$$
$$\ldots$$
$$(\exists y.E_{m,0}(y) \wedge f_{mi}(y) = x).$$

O caso mais simples de CUA é quando um evento tem somente um único tipo de evento como seu componente direto. Por exemplo, na figura 6, o fato que o agente está fervendo água, pode-se concluir que o agente está fazendo um prato de massa. Formalmente esta hipótese é que :

$$\forall x.ferve(x) \rightarrow \exists y.faz_prato_de_massa(y) \wedge x = f3(y)$$

Mais genericamente, a conclusão deste tipo de hipótese é a disjunção de todos os eventos que têm um componente que é compatível com a premissa. Considere a hipótese para faz_molho. Este tipo de evento é compatível com ele mesmo e todas as suas especializações: faz_molho_à_Perez, faz_molho_à_Helano, faz_molho_à_Furtado. A fórmula a seguir descreve todos os eventos que poderiam ter um componente dos tipos anteriores :

$$\begin{aligned}\forall x.faz_molho(x) \rightarrow\ & (\exists y.faz_prato_de_massa(y) \wedge x = f2(y)) \vee && \{1\}\\ & (\exists y.faz_fettucini_à_Perez(y) \wedge x = f2(y)) \vee && \{2\}\\ & (\exists y.faz_spaghetti_à_Helano(y) \wedge x = f2(y)) \vee && \{3\}\\ & (\exists y.faz_spaghetti_à_Furtado(y) \wedge x = f2(y)) \vee && \{4\}\\ & (\exists y.faz_bife_à_Furtado(y) \wedge x = f5(y)) && \{5\}\end{aligned}$$

Note que:

{1} é obtido pela lista de compatíveis
{2} é obtido pela verificação da lista de abstrações e seus componentes
{3}é obtido pela verificação da lista de abstrações e seus componentes
{4}é obtido pela verificação da lista de abstrações e seus componentes
{5}é obtido pela verificação da lista de abstrações e seus componentes

CUA pode ser visto como sendo um componente direto do evento observado ou um componente de um evento compatível ao evento observado. Note que faz_bife_à_Furtado tem como componente faz_molho_à_Furtado, que é compatível com faz_molho. A fórmula pode ser simplificada usando-se os axiomas de abstração de faz_prato_de _massa :

$$\begin{aligned}\forall x.faz_molho(x) \rightarrow\ & (\exists y.faz_prato_de_massa(y) \wedge x = f2(y)) \vee\\ & (\exists y.faz_bife_à_Furtado(y) \wedge x = f5(y))\end{aligned}$$

3.4 Cardinalidade Mínima (MCA)

As três hipóteses anteriores são formadas a partir da hierarquia de eventos antes que qualquer observação tenha sido feita. A hipótese de cardinalidade mínima (MCA) é baseada na hierarquia de eventos em observações as em um dado momento. Considere a seguinte seqüência de declarações :

MA_0 $\forall x.\sim fim(x)$
MA_1 $\forall x,y.fim(x) \wedge fim(y) \rightarrow x = y$
MA_2 $\forall x,y,z.fim(x) \wedge fim(y) \wedge fim(z) \rightarrow (x = y) \vee (x = z) \vee (y = z)$
. . .

A primeira assertiva diz que não existe nenhum evento do tipo "fim". A segunda, diz que existe somente um evento do tipo "fim". A terceira, diz que existe no máximo dois eventos do tipo "fim". E assim por diante.

Sejam estas observações apresentadas por um conjunto de fórmulas Γ. A hipótese de MCA para uma hierarquia H e um conjunto Γ é a fórmula MA_i; onde i é o *menor inteiro* tal que :

$\Gamma \cup EXA \cup DJA \cup CUA \cup MA_i$ é consistente.

Em geral, o teste de consistência é indecidível. Os algoritmos descritos na seção 4.3 criam estruturas de dados separadas correspondendo à aplicação de cada hipótese nas observações, em seguida é feito uma *poda* quando é notada uma inconsistência. As conclusões do sistema são representadas pela estrutura de dados correspondendo a hipótese mais forte. As hipóteses EXA, DJA e CUA não combinam as informações das diversas observações.

Suponha que o agente foi observado fazendo spaghetti e fazendo molho à moda Furtado. A primeira observação é explicada, por CUA dizendo, que o agente está fazendo spaghetti à moda Helano ou spaghetti à moda Furtado. A segunda observação é explicada , por CUA, e diz que o agente está fazendo spaghetti à moda Furtado ou bife à moda Furtado. Entretanto, não é possível concluir ao final que o agente está fazendo spaghetti à moda Furtado, pois o evento "fim" que explica a primeira observação pode ser diferente do evento "fim" que explica a segunda observação. A teoria destas observações permite especificar que :

$\exists x[faz_spaghetti_à_moda_Furtado(x) \vee faz_spaghetti_à_Helano(x)]$
$\exists y.[faz_spaghetti_à_moda_Furtado(y) \vee faz_bife_à_Furtado(y)]$

Em muitos casos, é correto assumir que as observações são relacionadas. Uma heurística a ser utilizada é que há um número mínimo de diferentes eventos do tipo "fim". No exemplo acima, todos os tipos são especializações de "fim". Logo, pelas observações acima, das hipóteses de disjunção e pela hipótese que não há mais que um evento do tipo "fim"; implica que o agente está fazendo spaghetti à moda Furtado.

4. Implementação do Reconhecedor

Muitos trabalhos têm sido desenvolvidos sem que haja um algoritmo ou mesmo uma implementação de reconhecimemento de planos. Dentre os poucos trabalhos que abordam o lado prático de reconhecimento vale a pena destacar os trabalhos de interfaces inteligentes — CHECS [GoLi91], o ATMS probabilístico — Wimp2 [GoCh88], o algoritmo de reconhecimento de planos para a teoria formal de Kautz [Kaut87] e o assistente inteligente encontrado em [LSP90].

O CHECS (Chemical Engeneering CAD System) emprega o modelo de reconhecimento de planos para encontrar os domínios dos planos que são usados pelo sistema. Ele é usado para tratar o conhecimento de processos químicos avisando ao projetista se um determinado projeto está correto, ou dando um projeto parcial que pode servir como complementação de vários projetos. A incorporação de um reconhecedor de planos dentro da intrface multimodal do CHECS é uma tentativa bem sucedida de aumentar o desempenho desta implementação.

Wimp2 é um programa feito para compreensão de estórias que as descreve em uma forma lógica, além de mostrar as inferências derivadas a partir das causas; tudo isto baseado em uma estrutura de planos. O Wimp2 utiliza a técnica de "forward chaining" com um sistema de contexto múltiplos — o ATMS (Assumption-based Truth Maintenence System). Com isso, ele pode declarar todas as possíveis inferências relevantes dentro da base de dados, refinando sua interpretação da entrada através da inclusão de restrições. O fator mais importante neste trabalho é o uso de probabilidades como uma forma de controle da busca dos planos candidatos à observação de uma determinada ação. Desta forma o espaço de estados possíveis é sensivelmente reduzido. Entretanto existem alguns problemas com o uso de probabilidades em sistemas reconhecedores. As probabilidades, fornecidas pelo os usuários, podem não ser bem atrabuídas; com isto teria que ser revisto todo o conhecimento para evitar que se tenha explicações que não correspondem com a realidade. Outro problema que pode aparecer é a renormalização dos pesos visando satisfazer os axiomas probabilísticos que têm que serem válidos a todo momento.

Um trabalho bastante prático no uso de geração e reconhecimento de planos pode ser encontrado em [LSP90]. Este trabalho trata o problema do desenvolvimento, implementação e manutenção de um assistente inteligente para indústria aeroespacial, mais especificamente a Companhia de Sistemas Aeronáuticos Lockheed. Para se conseguir tais objetivos é preciso uma interação entre sistema reconhecedor e usuário para o compartilhamento de conhecimento, pois o sistema pode interagir com o usuário visando o planejamento tático (planos) para o piloto, através de uma interface. Este sistema é capaz de dar suporte de informações, permitindo o piloto avaliar e executar o plano. Esta implementação também informa o planejamento tático da atividade do piloto, assim como os possíveis planos alternativos, sempre que é preciso uma tomada de decisão do piloto. Entretanto, este tarbalho tem alguns pontos falhos. Primeiro, não há um aprofundamento nos detalhes de implementação do sistema. Segundo, não é mostrado a parte formal do reconhecimento, caindo no antigo problema de reconhecimento de plano para casos específicos.

Apesar de Kautz não ter implementado um sistema reconhecedor de planos, é feito um estudo da teoria formal e baseado nesta teoria é apresentado um algoritmo de reconhecimento genérico. Este algoritmo aplica o reconhecimento de múltiplos planos, não somente de forma puramente dedutiva, mas trabalhando com toda forma de conhecimento do agente, seja ele explícito ou implícito. Com esta finalidade foram criadas diversas hipóteses heurísticas (EXA, DJA, CUA e MCA) que independem da aplicação, conforme visto no capítulo 3. Este algoritmo trabalha com domínios váriados, bastando modelar o conhecimento conforme a hierarquia de eventos. Desta forma ganha-se um algorimo genérico que pode, sem problema algum, ser refinado através da adição de novas heurísticas especificas para cada problema.

Em nosso trabalho, o processo de reconhecimento de planos foi codificado em linguagem PROLOG. Para isto criamos o SRP — Sistema Reconhecedor de Planos. O código fonte do SRP encontra-se no apêndice B.

4.1 Especificação do Sistema

Uma etapa importante na implementação do sistema de reconhecedor de planos é especificá-lo o melhor possível antes da sua codificação. Em problemas que envolve raciocínio este fato se torna crucial para atingir o objetivo final, pois nem sempre as informações passadas, pelo agente ao reconhecedor, são completas e bem elaboradas (de fácil captura da intensão).

Inicialmente será abordado os objetivos a serem alcançados pelo sistema de reconhecimento. Em seguida será visto a especificação funcional.

4.1.1 Objetivos

O objetivo principal do deste tarbalho é implementar o processo de reconhecimento de planos a partir da teoria formal de Kautz. Consoante citado na introdução, este processo pode trabalhar com as mais diversas formas de informação. Portanto, o reconhecedor a ser implementado deve trabalhar com informações completas ou incompletas. Além disso os planos recebidos como entrada podem ser totais ou parciais. Apesar da complexidade de se trabalhar com planos parciais, o reconhecedor conterá heurísticas que eliminarão estados do espaço de busca que venham a gerar redundâncias. Se o plano recebido pelo reconhedor for total e já estiver na forma de assertivas lógicas, então será verificado a biblioteca de planos comuns para procurar a existência de algum plano equivalente a ele. A partir deste passo será elaborada a explicação da ação ou plano que foi observado. Logo após a fase de explicação o programa elabora a saída (lista das derivações válidas e/ou explicação).

Por outro lado, se o plano dado pelo agente planejador ao reconhecedor for parcial, então será inicialmente criado um grafo de explicações (**e-grafo**), ou consultado se já existir. A partir da análise do e-grafo o módulo de análise do grafo tranforma esse grafo em assertivas lógicas. Neste momento o plano parcial segue os mesmos passos que o total. Vale a pena salientar que o processo de análise e implementação do reconhecimento de planos parcias envolve uma complexidade considerável, pois as informações passadas são incompletas.

Com o reconhecedor já pronto, este poderá dar respostas de maneira objetiva e satisfatória ao agente planejador que elaborou as assertivas de entrada. Desta forma o reconhecedor será capaz de inferir qual o estado inicial do planejador e então propor uma melhor maneira de solucionar o plano.

4.2.2 Epecificação Funcional

O Sistema Reconhecedor de Planos é uma ferramenta que usa diversas técnicas de raciocínio: racocínio por generalização, especificação e meta-nível. Portanto, existe uma dinâmica entre os seus predicados funcionais que deve ser tratada com cuidado durante a implementação. Uma visão da arquitetura do SRP pode ser vista na figura 7, onde o agente reconhecedor (homem ou máquina) observa um plano ou evento específico e a partir desta observação tenta descobrir a intensão do agente planejador (usuário). Para que isto possa ser feito o agente reconhecedor deve interagir com o Sistema de Reconhecimento de Planos — SRP. O SRP é composto de dois subsistemas básicos ou fundamentais: o gerador de explicações e o gerador de inferências. As explicações ou planos reconhecidos são obtidos a partir da hierarquia de eventos (modelo do conhecimento), das regras heurísticas e da Biblioteca de Planos Típicos — BPT. Também faz parte o subsistema de revisão de crenças que é o responsável pela assimilação e revisão das crenças existentes na hierarquia. Foi escolhido manter o módulo que possui as regras heuríticas em separado da hierarquia de eventos, pois facilita a manipulação e aquisição de novas regras. Também pode ser visto que o módulo

de revisão de crenças está representado de forma bastante simples. Isto deve-se a complexidade deste módulo, sendo portanto tratado detalhadamente em uma nova versão do SRP.

Alguns pontos importantes que não foram abordados em outras implementações serão tratados na implementação do SRP. Primeiro, com o SRP será possível tratar com objetos incorporados aos eventos e a partir da estrutura de dados criada podemos incorporar métodos de cálculo de atributos (campos ou facetas) que não estão previamente estabelecidos na base de conhecimento — chamados *procedimentos a estabelecer*. Segundo, será possível dar prioridades aos planos candidatos, que é de grande relevância em reconhecimento de planos. Assim é possível estabelecer uma preferência dentre qual explicação escolher como sendo a mais adequada para uma observação feita pelo agente. Por último, será feito o tratamento de exceções de eventos e objetos, desta forma evita-se que tenhamos uma representação do conhecimento incosistente com o domínio modelado.

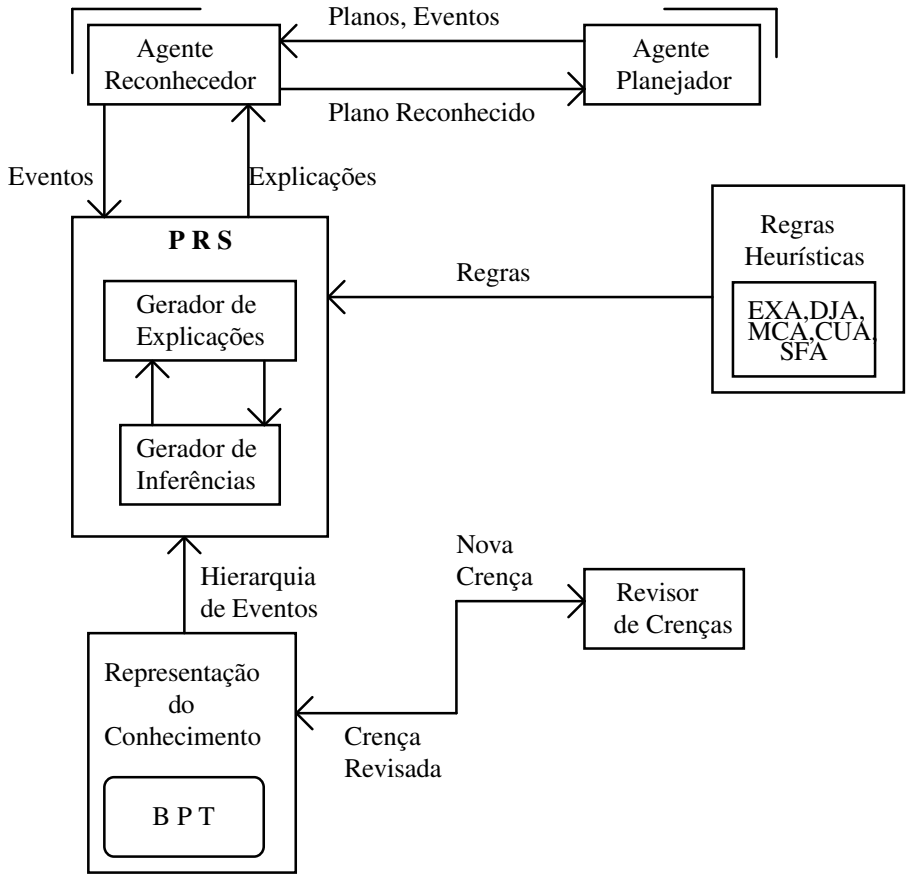

Figura 7 - Arquitetura do SRP

Após a implementação do reconhecedor ele poderá ser acoplado em diversas aplicações:

Banco de Dados Dedutivos:

Uma vez modelado o conhecimento ele pode ser armazenado em um banco de dados, então as consultas dos ususários podem ser otimizadas através do reconhecimento de seus planos. Toda vez que é efetuada uma consulta o reconhecedor é ativado para tentar deduzir os passos seguintes do usuário — suas intensões. Assim o Sistema Gerenciador de Banco de Dados Dedutivos (SGBDD) deduzirá as ações seguintes que são satisfatíveis com o contexto da consulta e caso o usuário faça uma consulta mal formulada ou incompleta será mostrado um aviso do erro junto com a explicação e suas alternativas de correção. Isto torna-se possível uma vez que o reconhecedor possui axiomas e regras que facilitarão esta tarefa. Note que o reconhecedor será, neste caso, um módulo de dedução do SGBDD.

Banco de Dados Temporais:

Um fator crítico em sistemas computacionais, sejam eles banco de dados ou não, é o tratamento das restrições temporais. O primeiro problema que aparece é a forma de representar o tempo em um banco de dados. A solução proposta vem da utilização de

algum formalismo que consiga representar o tempo de forma rigorosa e ao mesmo tempo expressiva. Diversas formas podem ser encontradas, porém, como foi visto na seção 2.3, uma boa alternativa para solucionar este problema é a álgebra de intervalos de Allen [Allen83], ou inda, conforme será adotado neste trabalho, a criação de relações em um nível mais alto, a partir das primitivas de Allen.

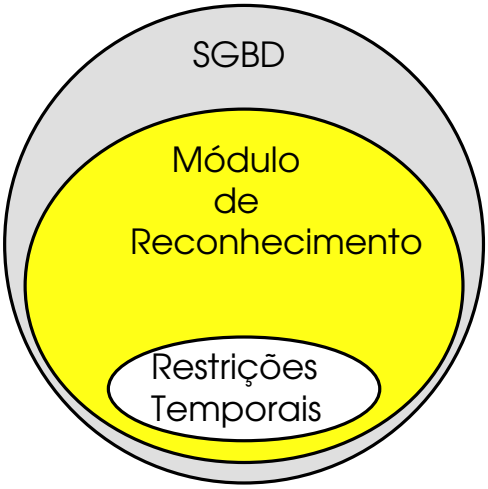

Figura 8 - Reconhecedor acoplado ao SGBD

Depois de organizado a estrutura de representação do tempo, surge o segundo problema, onde e como armazenar esta estrutura. Este problema de armazenamento das restrições temporais pode ser feito dentro do próprio módulo reconhecedor, que no caso faria parte do Sistema Gerenciado de Banco de Dados — SGBD.

Conforme pode ser encontrado na figura 8, o reconhecedor de planos ao fazer a busca dos planos candidatos já se encarrega de eliminar os planos do usuário que gerasse alguma inconsistência temporal, pois as restrições (temporais, ou não) estariam armazenadas nele. Entretanto, as restrições armazenadas no sistema podem diferir das restrições passadas pelo o usuário, então é o caso que se tenha uma interação entre o sistema de reconhecimento e o usuário. Se este o usuário tiver permissão para alteração na base temporal, então poderá efetivá-la através do próprio reconhecedor, do contrário o sistema deve fornecer um aviso explicativo sobre os motivos da inconsistência temporal e ao mesmo tempo sugerir os procedimentos necessários para uma possível mudança na base de dados.

Um outro problema que pode surgir com BD temporais é quando este permite consultas utilizando linguagem natural. O reconhecedor poderá tratar este problema através da incorporação de um procedimento que captura as restrições temporais a partir dos tempos dos verbos. Um algoritmo proposto para esta finalidade pode ser encontrado em [CSSB91], onde é feito uma análise temporal a partir de uma hierarquia dos tempos destes verbos.

Interfaces Inteligentes:

Uma boa forma de utilização de reconhecimento de planos encontra-se no uso em interfaces inteligentes. Já existem alguns trabalhos ocorrendo neste sentido, um bom exemplo pode ser encontrado em [GoLi91]. Neste trabalho foi construída uma ferramenta baseada em planos onde o plano do usuário é inferido de acordo com sua entrada pela interface multimodal.

A aplicação de reconhecimento de planos pode ocorrer em todos os tipos de interfaces. As interfaces orientadas por comandos podem beneficiar-se usando o reconhecedor de planos para consertar erros de sintaxe de uma solicitação; inferindo o objetivo do usuário a interface anteciparia os comandos seguintes e, desta forma, forneceria o contexto necessário para recuperar o comando mal formulado. As interfaces gráficas, por sua vez, necessitam a manipulação de ícones na tela para descobrir da intensão do usuário. O reconhecimento destas intensões por um reconhecedor de planos pode ajudar a garantir que os ícones mais relevantes estarão disponíveis em cada tela. No caso de interfaces baseadas em linguagem natural, o reconhecedor tem um papel de grande importância, pois neste tipo de interface o usuário pode submeter solicitações sem sentido ao sistema, o que acontece com uma relativa freqüência. Assim, o reconhecedor de planos poderá gerar detecção de contexto e preencher a falta de informações deixada pelo usuário. Por último, para o uso em interfaces multimodais o sistema reconhecedor de planos torna-se requisito básico e essencial, pois este tipo de interface incorpora todas as dificuldades dos outros estilos. Além disso, há uma necessidade extra ao fazer o controle de qual é o melhor tipo de interface a ser utilizada pelo usuário a cada tela.

Assistentes Inteligentes:

Assistentes inteligentes são sistemas de computação que permitem a cooperação entre homem e máquina. A necessidade de tais sistemas já se torna fundamental em diversos sistemas complexos, provendo o auxílio necessário ao usuário na manipulação destes sistemas. Estes sistemas podem ser encontrados nas mais diversas aplicações — sistemas de controle de aviões, ônibus espaciais e veículos sofisticados. Em [LSP90] pode ser encontrado em destes assistentes inteligentes que foi produzido para Companhia de Sistemas Aeronáuticos Lockheed. Neste sistema o reconhecedor seleciona um plano da sua biblioteca de planos, de acordo com a circunstância enfrentada pelo piloto. O reconhecedor então sugere este plano ao piloto via interface piloto-veículo, este por sua vez aprova ou não o plano sugerido. Ademais, o reconhecedor pode fornecer todo tipo de informações sobre o plano tático do piloto e a cada momento antecipar e ajudar nas futuras ações deste piloto. Caso o piloto necessite informações que expliquem os motivos pelo qual o assistente escolheu um determinado plano, este assistente em interação com seu módulo reconhecedor, deverá fornecê-las sem nenhuma complicação. O trabalho de [LSP90] torna-se básico para este tipo de aplicação de reconhecimento, pois é feito a ligação entre as duas sub-áreas de planejamento. Desta forma o reconhecedor prover ajuda ao usuário, sugerindo todo tipo de assitência que ele possa precisar na elaboração e execução de seu plano tático.

Hipertextos:

Sistemas de hipertextos trabalham com ligações que armazenam as consultas e o contexto das solicitações do usuário. É comum ouvir usuários de sistemas hipertexto dizendo que se perderam durante sua navegação devido a quantidade de âncoras do sistema. O sistema de reconhecimento de planos poderá fornecer um auxílio importante em tais consultas do usuário, pois a cada solicitação seria mostrado os seus planos alternativos naquele contexto específico. Assim o reconhecedor traria para memória somente estes planos canditatos a solução do seu problema. Desta forma, seria evitado o problema do usuário se perder em suas consultas, visto que sua árvore de busca de informação está sendo mostrada a cada solicitação. Caso o sistema hipertexto incorpore muitas informações e ligações, o que geralmente ocorre, então a solução mais realística é armazenar somente os planos candidatos mais prováveis à resposta da solicitação do usuário e, se o usuário solicitar ajuda, o sistema reconhecedor dará as informações para que o usuário não se perca no hipertexto. O maior problema surge quando tem que se dar preferência sobre alguns planos possíveis, porém isto pode ser resolvido através de um escolha feita pelo próprio usuario, que tem em sua tela estes planos, sendo ele então o encarregado da atribuição de preferência.

Re-Engenharia de Software :

O reconhecedor de programas é uma técnica, dentre várias, usada pelos programadores na tarefa de entendimento de programas. Sabendo como as estruturas de código são implementadas, ou especializadas de estruturas mais abstratas, o programador pode construir uma descrição hierarquica do projeto do programa que esta sendo examinado. Assim, o reconhecedor encontra estas estruturas básicas e a partir delas controi os termos básicos e suas derivações, facilitando o reconhecimento automático dos programas. Para isto, o reconhecedor deve ser capaz de verificar o grafo do programa e prover uma descrição do projeto de forma coesa e simplificada. Uma abordagem prática deste tipo de utilização do reconhecimento de planos pode ser encontrado em [Will 90].

Recuperação de Texto:

Uma forma rápida e inteligente de recuperação de textos pode ser feita utilizando-se um reconhecedor de planos. Inicialmente o texto deverá ser composto em uma forma estrutural. Ou seja, o texto deve ser mapeado em estruturas de forma a representar sintática e semanticamente o texto, isto pode ser elaborado a partir de um gerador de planos e logo após esta etapa este gerador se encarrega de mapear estas estruturas em planos. Uma vez transformado o texto em estrutura, o usuário poderá fazer a recuperação deste texto, ou ainda de vários textos. A partir do reconhecimento destas estruturas, utilizando como partida a intensão do usuário, o reconhecedor poderá fazer a comparação de textos similares ou que venha a ter ligações com o texto que está sendo procurado.

Note que no caso de hipertexto estas ligações já existem e para o caso de recuperação de textos esta tarefa é bem mais complexa, pois as ligações inicialmente não

existem e são construídas a partir da descoberta da intensão do usuário. Portanto, basta o uso do reconhecedor de planos na busca pela descoberta da intensão do usuário. Uma vez descoberto o plano (intensão) do usuário o reconhecedor trará para memória tanto os textos solicitados quanto os ponteiros que apontam para todos os textos identificados como sendo equivalentes ou como tendo relação com o texto solicitado.

A identificação do texto a partir de estruturas torna-se um ponto importante para usuários ou pesquisadores que trabalham com áreas afins, pois a partir de um texto pode-se descobrir textos relacionados e mesmo similares que passavam despercebidos por tais usuários. Além disso, o reconhecedor será capaz de dar informações e explicações de porque reconheceu um texto como sendo similar ou relacionado com outro.

Otimização de Tradutores:

Uma gramática e suas produções podem ser mapeadas em uma estrutura de hierarquia de eventos, onde seriam representadas as abstrações, especializações e decomposições. Logo, o reconhecedor poderá ser utilizado com a finalidade de detectar e corrigir erros no momento de codificação de um programa. Outro uso mais realístico e prático é a incorporação do reconhecedor como sendo um módulo de tradutores. Desta forma, a detecção de erros ficaria sob responsabilidade do tradutor e o reconhecedor teria como finalidade a correção dos erros e geração de formas alternativas de estrutura melhores para o programador. Todavia, o processo de melhoramento do programa do usuário deve ser feito de modo cooperativo, ou seja, o programador poderá, de acordo com a sua prioridade de projeto, incluir novas estruturas gramaticais, excluir estruturas, ou ainda alterar a biblioteca de planos. Desta forma o programador pode verificar seus erros ou estruturas mal formuladas e pedir ajuda e explicações ao reconhecedor.

5. Análise da Complexidade

O processo de reconhecimento de planos não é um problema trivial. Na verdade, este processo é complexo e despendioso, pois o conhecimento do agente está representado junto com todas as interrelações entre os eventos formadores deste conhecimento. Deve ser lembrado que os axiomas e hipóteses heurísticas manipulam este conhecimento através da hierarquia de eventos. Desta forma, o reconhecimento de planos tem uma complexidade baseada na combinação dos eventos passados como entrada nesta hierarquia de eventos.

Não obstante, existem maneiras de reduzir a complexidade do problema fazendo-se algumas modificações no trabalho de Kautz. Em [Vilain90], o processo de reconhecimento é feito usando-se gramática livre de contexto e a partir desta gramática utilizam-se algoritmos que tratam a complexidade do problema de forma reduzida — complexidade polinomial. Assim, é utlizado uma abordagem de "parsing" para produzir estratégias de reconhecimento com melhor desempenho do que as desenvolvidas por Kautz. Todavia, a abordagem de Vilain faz bastante restrições ao modelo inicial proposto por Kautz. Dentre estas restrições vale a pena destacar: em seu paradigma os passos de decomposição não são sobrepostos, não trabalha com planos parciais, os

planos são considerado como estando totalmente ordenados e por último não trata multidomínios. Vilain também prova que o processo de reconhecimento de planos com abstrações e planos parciais é um problema NP completo.

O algoritmo de reconhecimento é no pior caso exponencial no tamanho da entrada da hierarquia de eventos, pois um evento pode ter diversos componentes diferentes e que possuem o mesmo tipo. Na prática, o primeiro passo do algoritmo de reconhecimento geralmente encontra um nó similar e poda a pesquisa durante a busca.

Apesar da teoria proposta ser completa os algoritmos são incompletos, pois na teoria existem conjuntos que falam de eventos que não aparecem na hierarquia de abstração (H_G) e que não são tratados em nenhum algoritmo proposto. Entretanto, a teoria especifica um conjunto de axiomas infinitos, na prática os algoritmos especificam somente as conclusões que podem ser computadas. Também são usadas estruturas de dados mais poderosas como alternativa de reduzir a complexidade do problema de reconhecimento de planos.

6. Conclusão

Este trabalho teve como objetivo fazer uma análise detalhada da teoria de reconhecimento de planos, bem como formas de implementá-la de forma eficiente. Foi adotado, dentre as diversas teorias existentes [Kaut87, CSSB91, Wobc88, KoPo89], a teoria de Kautz como base para implementação do reconhecedor. Diversos exemplos foram formulados com a finalidade de se obter praticidade e relevância no uso do reconhecedor de planos.

Foram desenvolvidos métodos e esquemas para o processo de reconhecimento de planos, que estão incluídos em duas partes. Na primeira, teórica, foram abordadas as diversas formas de representar o conhecimento do agente, assim como a linguagem utilizada para este processo, o tratamento temporal e as hipóteses heurísticas. Na segunda parte, prática, trabalhou-se formas de se implementar toda a teoria proposta no trabalho. Por outro lado, procurou-se manter todas as etapas da criação de um sistema computacional — especificação e projeto modular. Apesar de toda complexidade do problema foi conseguido resultados bem genéricos para o reconhecedor de planos :

- Os planos com que o reconhecedor trabalha podem ser totais ou parciais
- Pode tratar informações completas e incompletas
- Não assume que todos os passos do plano então lineamente ordenados, ou seja, reconhece planos lineares ou não.
- Trata as retrições contidas nos fatos da base de conhecimento
- Trata as restrições temporais entre os eventos

- Trata as restrições de igualde entre os objetos

Apesar dos objetivos alcançados, este processo possui ainda algumas limitações tais como : não tratar revisão de crenças ou aprendizado, o reconhecedor vê somente um evento do tipo FIM de cada vez e por último não trabalha com herança múltipla nos axiomas de abstração.

Apêndice A

Prova das Hipóteses Heurísticas

As três proposições a seguir demonstram que as classes de hipóteses — exaustividade, disjunção e componente/uso, são equivalentes à circunscrições particulares da hierarquia de eventos. A prova de cada uma delas se basea em duas estratégias :

- (Estartégia 1) Para provar que algum conjunto de fórmulas $S \cup T$ implica Circum(S, π), basta mostrar que todo modelo de $S \cup T$ é minimal em π relativo a S.

- (Estartégia 2) Para provar que Circum(S,π) implica em algum conjunto de fórmulas $S \cup T$, basta mostrar que T vale (é satisfatível) em todo modelo minimal em π relativo a S.

Proposição 1 :

Declara que as hipóteses de exaustividade (EXA) são obtidas por circunscrição dos tipos de ventos não básicos da hierarquia de abstração.

$$\mathbf{H_A \cup EXA \Leftrightarrow Circum(H_A, H_E - H_{EB})}$$

Prova 1 :

$$\mathbf{Circum(H_A, H_E - H_{EB}) \Rightarrow H_A \cup EXA}$$

Hipótese 1 : Suponha que $\{E_1, \ldots, E_n\}$ são todos os predicados diretamente abstraídos por E_0 em H_A. Ou seja, suponha que a hipótese de exuastividade é verdadeira.

Queremos que a proposição $\forall x.E_0(x) \Rightarrow (E_1(x) \vee \ldots \vee E_n(x))$ que seja verdadeira em todos os modelos de H_A que são mínimos em $H_E - H_{EB}$.

Seja M_1 um modelo minimal de H_A em que a proposição EXA não vale.

Logo, deve existir algum :C tal que $E_0(x) \wedge \sim E_1(x) \wedge \ldots \wedge \sim E_n(x)$ seja verdadeiro em M_1 {x/:C}.

Definimos um modelo M_2 por :

$Dom(M_2) = Dom(M_1)$
$M_2[Z] = M_1[Z]$, para $Z \neq E_0$
$M_2[E_0] = M_1[E_0] - \{:C\}$

Ou seja, M_2 é o mesmo que E_1 exceto que $:C \notin M_2[E_0]$. {prova por RAA}

Provaremos que $\forall x.E_i(x) \Rightarrow E_0(x)$, para $1 \leq i \leq n$, são falsos em M_2 somente se existe um :D tal que $:D \in M_2[E_i] \wedge :D \notin M_2[E_0]$

Se $:D \neq :C$
Então $:D \in M_2[E_i] \Rightarrow$
$:D \in M_1[E_i] \Rightarrow$
$:D \in M_1[E_0] \Rightarrow$
$:D \in M2[E_0]$, que é uma contradição !
Senão $(:D = :C)$
$:D \in M_2[E_i] \Rightarrow$
$:D \in M_1[E_i] \Rightarrow$
$:D \in M_1[E_0] \Rightarrow$
$:C \in M_1[E_0]$ M1[E_0], que é uma contradição !

Conclusão: Portanto, não pode existir :D, logo M_2 é um modelo de H_A e M_1 não é mínimo.■

➲

$$\mathbf{H_A \cup EXA \Rightarrow Circum(H_A, H_E - H_{EB})}$$

Primeiro temos que provar o seguinte lema:

(Lema 1) Todo evento em modelo de $H_A \cup EXA$ é de no *mínimo* um tipo básico. Isto é, se M_1 é um modelo de $H_A \cup EXA$ tal que $:C \in M_1[E_0]$, então existe um tipo de evento básico $E_b \in H_{EB}$ tal que $:C \in M_1[E_b]$ e E_0 abstrai = E_b

(Prova do lema) - por indução :

Definimos uma ordem parcial sobre H_E , $E_J < E_k$ sss E_k abstrai E_J.

Suponha que o lema é válido para todo $E_i < E_0$.

Desde que $\forall x.E_0(x) \Rightarrow (E_1(x) \vee \ldots \vee E_n(x))$ e $:C \in M_1[E_0]$; deve ser o caso que $:C \in M_1[E_1] \vee \ldots \vee :C \in M_1[E_n]$.

Suponha que $:C \in M_1[E_1]$, então existe E_i abstrai = E_b e $:C \in M_1[E_b]$. Desde que E_0 abstrai = E_1 e também abstrai = E_b, isto completa a prova do lema■

Provamos que se M_1 é um modelo de $H_A \cup EXA$, então M é um modelo de H_A minimal em .$H_E - H_{EB}$.

Suponha que não (M não é um modelo de H_A minimal em $H_E - H_{EB}$).

Então existe um M_2 tal que $M_1 >> M_2$, existe (no mínimo um) $E_0 \in H_E - H_{EB}$ e existe um evento :C tal que $:C \in M_1[E_1] \wedge :C \notin M_2[E_0]$.

Pelo lema 1, há um $E_b \in H_{EB}$ tal que $:C \in M_1[E_b]$ Desde que M_1 e M_2 são válidos em H_{EB} , $C \in M_2[E_b]$, (pois E_0 abstrai = E_b), então $:C \in M_2[E_0]$ que é uma contradição. Portanto, não pode existir M_2.■

Proposição 2 :

Declara que as hipóteses de disjunção (DJA) são obtidas pela circunscrição de todos os evetos do tipo TOTAL, resultando um conjunto de fórmulas. Esta circunscrição garante que cada evento "token" tem no máximo um evento do tipo básico.

$$\mathbf{H_A \cup EXA \cup DJA \Leftrightarrow Circum(H_A \cup EXA, H_E - \{Evento\ TOTAL\})}$$

Prova 1 :

$$\mathbf{Circum(H_A \cup EXA, H_E - \{Evento\ TOTAL\}) \Rightarrow H_A \cup EXA \cup DJA}$$

Queremos que ,se os eventos E1 e E2 não são compatíveis então a declaração $\forall$ x.~E1(x) $\vee$ ~E2(x) é verdadeira em todos os modelos de $H_A \cup$ EXA que são mínimos em $\mathbf{H_E}$ **- {Evento TOTAL}**

Seja M1 um modelo "minimal" de $H_A \cup$ EXA em que a declaração seja falsa e :C seja um evento tal que E1(X) $\wedge$ E2(x) seja verdadeiro em M1{x/:C}.

Usando lema da prova da proposição 1, seja Eb um evento do tipo básico abstraído por E1 tal que :C $\in$ M1[Eb]

Definimos M2 como :
dom(M2) = dom(M1)
M2[Z] = M1[Z] para Z $\neq H_E$ /* é um evento que não participa de H_E */
M2[Ei] = M1[Ei] se Ei abstrai = Eb
M1[Ei] - {:C}, caso contrário

Note que M1[Evento TOTAL] = M2[Evento TOTAL], desde que Evento TOTAL certamente abstrai Eb.

Queremos que M2 seja um modelo de $H_A \cup$ EXA

(Prova que M2 é um modelo de H_A)

Suponha que não, ou seja, Suponha que o axioma $\forall$x.Ej(x) $\rightarrow$ Ei(x) seja falso em M2. Desde que ele é verdadeiro em M1 e M2 difere de M1 somente na ausência de :C da extensão de algum evento; deve ser o caso que :C $\in$ M2[Ej] e :C $\notin$ M2[Ei], enquanto :C $\in$ M1[Ej] e C $\in$ M1[Ei].

Por definição de M2 Ej abstrai = Eb

Desde que Ei abstrai Ej então Ei abstrai = Eb, mas M1 e M2 terão que equivalerem (concordar) em Ei, isto é, :C ∈ M2[Ei], que é uma contradição ! ■
/*pois Ei e Ej abstraem Eb */

(Prova que M2 é um modelo de EXA)

Suponha que não,

$\forall x.Ej_0(x) \Rightarrow (Ej_1(x) \vee \ldots \vee Ej_n(x))$ é falso.

Então deve ser o caso que

$:C \in M2[Ej_0] \wedge :C \notin M2[Ej_1] \wedge \ldots \wedge :C \notin M2[Ej_n]$

Mas $:C \in M2[Ej_0]$ significa que Ej_0 abstrai = Eb. Já que Ej_0 não é básico , no mínimo um $Ej_1, \ldots Ej_n$ abstrai = Eb

Suponha que Ej_1 abstrai = Eb
Então $:C \in M1[Eb] \Rightarrow :C \in M2[Ej_1]$, que é uma contradição !

Note que : Como E1 e E2 não são compatíveis, E2 não pode abstrair = Eb. Assim $:C \notin M2[E2]$. Logo M1 e M2 diferem no mínimo em E2. Portanto M1 >> M2, logo M1 não é mínimo.■

Prova 2 :

➲

$H_A \cup$ EXA $\cup$ DJA $\Rightarrow$ Circum($H_A \cup$EXA, H_E - {Evento TOTAL})

Lema 2 :

Todo evento em um modelo de **$H_A \cup$ EXA $\cup$ DJA** é de exatamente um tipo básico. Pelo lema da proposição 1, há no mínimo um tipo básico e por DJA nenhum evento é de dois tipos básicos.

Prova lema-2 :

Suponha que existe um M2 tal que M1 >> M2 e existe (no mínimo um) $Eo \in H_E$ - {Evento TOTAL} e um evento :C tal que $:C \in M1[Eo]$ e $:C \notin M2[Eo]$
Então $:C \in M1[Eo] \Rightarrow :C \in M1[$Evento TOTAL$]$
$:C \in M2[$Evento TOTAL$]$

Pelo lema da proposição 1 há algum $Eb \in H_A$ tal que $:C \in M2[Eb]$.

Prova da Proposição :

Desde que M1 >> M2, deve ser o caso que :C ∈ M1[Eb]. Pelo pelo lema acima. Eb é o único tipo básico de :c em M1 e Eo abstrai = Eb. Mas Eo abstrai = Eb significa que :C ∈ M2[Eb] ⇒ :C ∈ M2[Eo], que é uma contradição !

Portanto, não pode haver M2, logo M1 deve ser mínimo

Proposição 3 :

Intuitivamente, eventos do tipo FIM "apenas aparecem" e não são explicados pelas ocorrências deles como passo de algum outro evento. Minimalizando todos os eventos não FIM significa que os eventos ocorre somente quando são FIM, ou um passo de FIM, ou um passo de um passo de FIM. Ou seja, um evento Ei implica a disjunção de todos os eventos que poderiam ter o Ei como componente.

A terceira hipótese declara que CUA resulta da circunscrição de todos os eventos não FIM relativo a hierarquia de eventos junto com EXA e DJA.

$$\mathbf{H_A \cup EXA \cup DJA \cup CUA \Leftrightarrow Circum(H \cup EXA \cup DJA, H_E - \{Evento\ FIM\})}$$

Prova 1 :

$$\mathbf{Circum(H \cup EXA \cup DJA, H_E - \{Evento\ FIM\} \Rightarrow H_A \cup EXA \cup DJA \cup CUA}$$

Primeiro dev-se provar o seguinte lema:

Lema 3 :

Suponha que M1 é um modelo de $\mathbf{H_A \cup EXA \cup DJA}$ que é minimal em H_E - {Evento FIM}

Se :C1 ∈ M1[E1], para qualquer evento E1

Então :C1 ∈ M1[FIM] ou exite alguma produção :C2 tal que C1 é um componente direto de :C2

Prova do Lema :

Suponha que o lema acima é falso, (há eventos que ocorrem e que não são FIm nem um passo de FIM)

Define-se M2 como sendo :

M2[Z] = M1[Z] para $Z \neq H_E$ /* é um evento que não participa de H_E */
M2[Ei] = M1[Ei] - {:C1}, para todo $Ei \in H_E$

Note que M1 e M2 equivalem (concordam) em FIM

Será mostrado que $H \cup EXA \cup DJA$ são válidos em M2. isto significa que M1 >> M2, que é uma contradição.

Considere cada um dos tipos de axiomas de H:

H: $<H_E, H_A, H_{EB}, H_D, H_G>$

Caso 1 :

Axiomas em H_G devem valer, pois eles recebem mesma valoração em M1 e M2.

Caso 2 :

Axiomas em H_A são da forma :

$\forall x.Ej(x) \rightarrow Ei(x)$

Suponha que é falso. Então para algum :D, $:D \in M2[Ej] \wedge :D \notin M2[Ei]$
Porém isto é impossível, pois M1 e M2 de se equivaler quando $:D \neq :C1$, como deve ser o caso porque :C1 não aparece na extensão de qualquer tipo de evento de M2.

Caso 3 :

Axiomas em EXA são válidos pelo mesmo argumento acima.

Caso 5 :

O j-ésimo axioma em H_D é da forma :

$\forall x.Ej_0(x) \Rightarrow (Ej_1(fj_1(x)) \wedge Ej_2(fj_2(x)) \wedge \ldots \wedge Ej_n(fj_n(x)) \wedge \kappa$

Suponha que esse axioma não é válido em M2.
Então deve existir algum :C2 tal que

$Ej_0(x) \wedge \{\sim Ej_1(fj_1(x)) \vee \ldots \vee \sim Ej_n(fj_n(x))\}$ é válido em M2{x/:C}

M1 e M2 equivalem-s em κ. Assim deve ser o caso que para algum j e i, $M2[Eji](:C2) \notin M2[Eji]$,enquanto $M1[Eji](:C2) \in M1[Eji]$. Pois m1 e M2 diferem em Eji somente em :C1, então deve ser o caso que M1[Eji](:C2) = :C1. Mas então :C1 é um componente de C2 em M1, contrário hipótese original. Isto completa a prova do Lema.

Considere um membro arbitrário de CUA como definido anteriomente e que tem um predicado E do seu lado esquerdo.

Seja M um modelo de H $\cup$ EXA $\cup$ DJA que é minimal em H_E - {Evento FIM} tal que :C $\in$ M[E] e :C $\notin$ M[FIM]. Pelo lema acima, existe um Ejo, Eji tal que :

:D $\in$ M[Ejo]
:C $\in$ M[Eji]
:C = M[fji](:D)

onde fji é uma funçào em um axioma de decomposição para Ejo. Já que M é um modelo de DJA; e E e Eji são compatíveis. É verdadeiro afirmar em :C $\in$ M[Eji] que M é instanciado em :C, pois a disjunção contendo Eji é verdadeira quando a vairavél y é instanciada em :D. Desde que a escolha de M foi arbitrária, aformula de entrada é verdadeira em M.

Finalmente , desde que a escolha de M e o membro de CUA foi arbitrária, tudo de CUA é verdadeiroem todos os modelos de H $\cup$ EXA $\cup$ DJA que são mínimos em H_E - {FIM}.

Bibliografia

[Alle83] Allen J. F. , "Recognition Intentions from Natural Language Utterances", Computational Models of Discurse, m. Brady ed, MIT Press, Cambridge, Ma., 1983

[ChSh94] Chaniak E. e Shimony S.E. "Cost-Based Abduction and MAP Explanation", Artificial Intelligence Journal, volume 66, número2, 1994

[CoFe82] Cohen P. R. e Feigenbaum E. A. "The Handbook of Artificial Intelligence", volume 3, 1982

[CSSB91] Cohen R. , Song F. , Spencer B. e Beek P. V. "Exploiting Temporal and Novel Information from the User in Plan Recognition", User Modeling and User-Adapted Interaction, volume 1, 1991

[Dale89] Dalen, D.V. "Logic and Structure", Springer Verlag, second edition, 1989

[Ende73] Enderton H. B. "A Mathematical Intoduction to Logic", Academic Press Incorporation, second edition, 1973

[Ghal93] Ghallab M. "Maning temporal Knowledge for Interpretatation and Planning", X Simpósio Brasileiro em Inteligência Artificial, 1993

[Gins87] Ginsberg M. L. "Readings in Nonmonotonic Reasoning", Morgan Kaufmann Publishers, Ca., 1987

[GoLi91] Goodman B. A. e Litman D. "On the Interaction between Plan Recognition and Intelligent Interfaces", User Modeling and User-Adapted Interaction, volume 1, 1991

[GoCh88] Goldman R. e Charniak E. "A probabilistic ATMS for plan Recognition", AAAI-88 Workshop on Plan Recognition, 1988.

[Kaut87] Kautz H.A. "A Formal Theory of Plan Recognition", Morgan Kaufman, 1987.

[KaPe88] Kautz H.A. e Pednault, P.D. "Planning and Plan Recognition", AT&T Techenical Journal,volume 67, 1988

[KoPo89] Konolige K. e Pollack M. E. "Ascribing Plans to Agents", Eleventh International Joint Conference on Artificial Intelligence, Detroit, Mi., 924-930, 1989

[LSP90] Levi K. R. , Shalin, V. L. e Perschbacher D. A. "Learning Plans for an Intelligent Assistant by Observing User Behavior", International Journal of Man-Machine Studies, volume 33, 1990

[Luka90] Lukasziewiczs W. "Non-Monotonic Reasoning Formalizations of Common Sense Reasoning", Ellis Horwood Limited, 1990

[Matt91] Mattos N. M. "An Aproach to Knowledge Base Management", Lectures Notes in Artificial Intelligence, Springer-Verlag, 1991

[McCa80] McCarthy J. "Circumscription - Formal of Non-Monotonic Reasoning", Artificial Intelligence Journal, 1980.

[McCa86] McCarthy J. "Aplications of Circumscription to Formalizing Common Sense Knowledge ", Artificial Intelligence Journal ,volume 28, 1986

[Nils80] Nilson N. J. "Principles of Artificial Intelligence", Tioga Publish, 1980

[Pere92] Perez J. J. A. "Planejamento: uma revisão", Relatório Técnico, DI-PUCRJ, 1992

[Reit87] Reiter, R. "Non-Monotonic Reasoning", Annual Review of Computer Science, 1987

[Vila90] Vilain M. "Getting Serious about Parsing Plans: a Grammatical Analisys of Plan Recognition", AAAI-90, Proceedings Eighth Conference National on Artificial Intelligence, MIT Press, Cambridge, MA., 1990

[Will90] Wills L. M. "Automated Program Recognition : A Feasibility Demonstration ", Artificial Intelligence Journal, volume 45, 1990

[Wins84] Winston P. H. "Artificial Intelligence", Addison-Wesley Publish Company, second edition, Ma., 1984

[Wobc88] Wayne W. "A Logical Theory of Plan Recognition", AAAI-88, Workshop on Plan Recognition,1988