Planejamento em IA

Blackbox
Graphplan + SATplan

Leliane Nunes de Barros
DCC - IME - USP

Grafo de Planejamento [Blum&Furst 1997]

- Grafo de níveis: seqüência de níveis que correspondem a passos de tempo no plano. Nível 0 é o estado inicial.
  - nível *i*: contém um conjunto de proposições
  - nível *i+1*: contém um conjunto de ações

Nível *i* contém todos os literais que *poderiam* ser verdadeiros no passo *i* , dependendo das ações executadas nos passos $j > i$

Grafo de planejamento

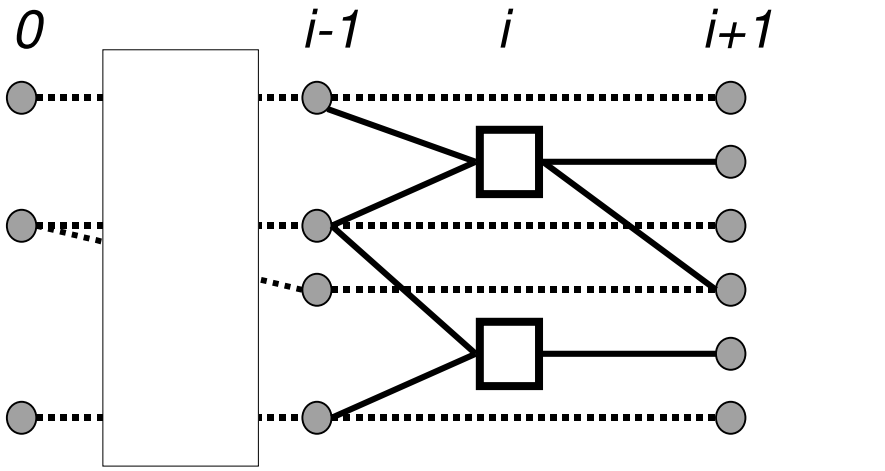
- O número de passos de um grafo de planejamento fornece uma boa estimativa da dificuldade de se atingir um dado literal a partir do estado inicial
- Funciona apenas para problemas de planejamento proposicionais (sem variáveis)
- O grafo de planejamento pode ser construido de modo eficiente para problemas difíceis (muitos objetos resultando em muitas ações)

GRAPHPLAN [Blum&Furst 1997]

- O planejador *Graphplan* alterna entre duas fases:
  - expansão do grafo de planejamento: extende o grafo progressivamente até atingir uma condição necessária (mas não suficiente) para um plano existir.
  - extração da solução: faz uma busca regressiva no grafo a procura de um plano solução; se não o encontrar, *Graphplan* volta a expandir o plano.

Grafo de Planejamento

Expansão do Grafo de Planejamento



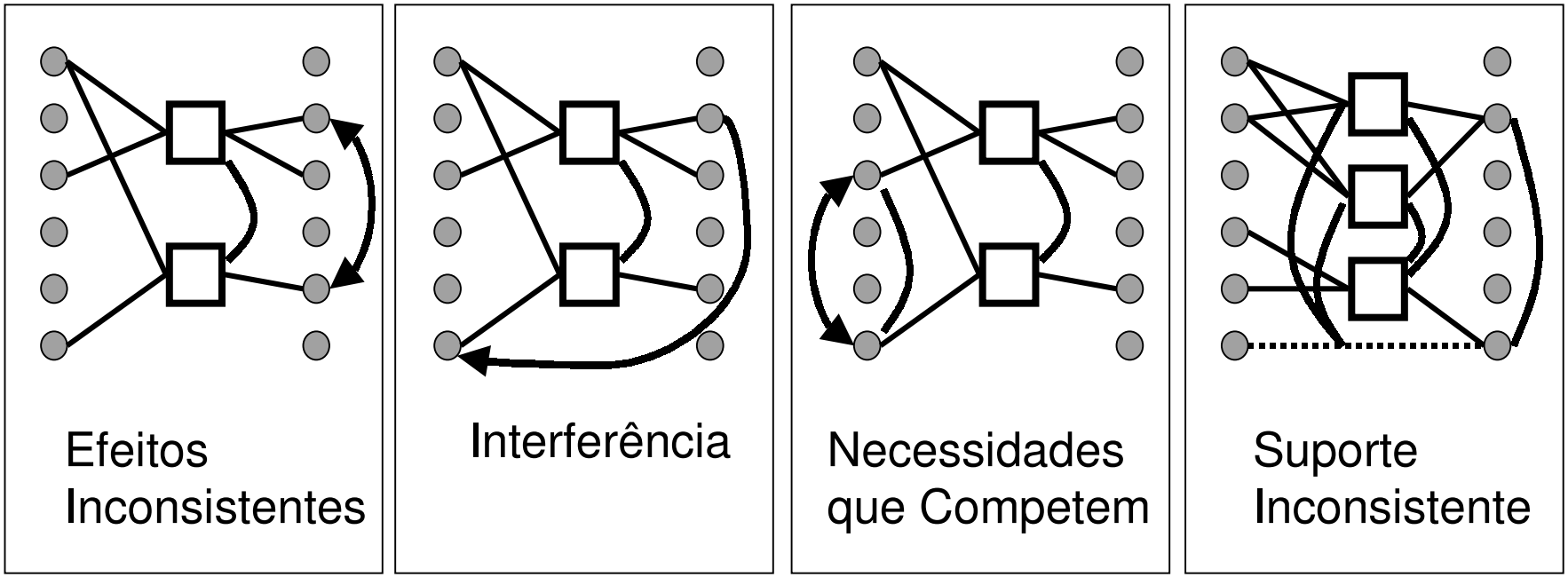
Leliane Nunes de Barros
DCC - IME - USP

Grafo de Planejamento

- Funciona apenas com ações proposicionais e estados completos
- O processo de construção do grafo não envolve escolha de ações (busca combinatorial) mas registra ações que não podem ser escolhidas com ligações *mutex* (*mutual exclusion*)
- A construção do grafo é polinomial no número de ações e proposições
- ações “paralelas” num mesmo nível não significa que elas serão executadas ao mesmo tempo.

Grafo de Planejamento

- Relações de Mutex (Exclusão Mútua - arcos em amarelo)



Ligações de *mutex* entre proposições

- Suporte inconsistente: Duas proposições *p* e *q* no nível *i* são mutex se (1) *p* é a negação de *q ou* (2) se todas as maneiras de se atingir *p* e *q* são mutex

Problema: “Have a cake and eat cake too”

Init (*Have*(*Cake*))

Goal (*Have*(*Cake*) $\wedge$ *Eaten*(*Cake*))

Action (*Eat*(*Cake*)

Precond: *Have*(*Cake*)

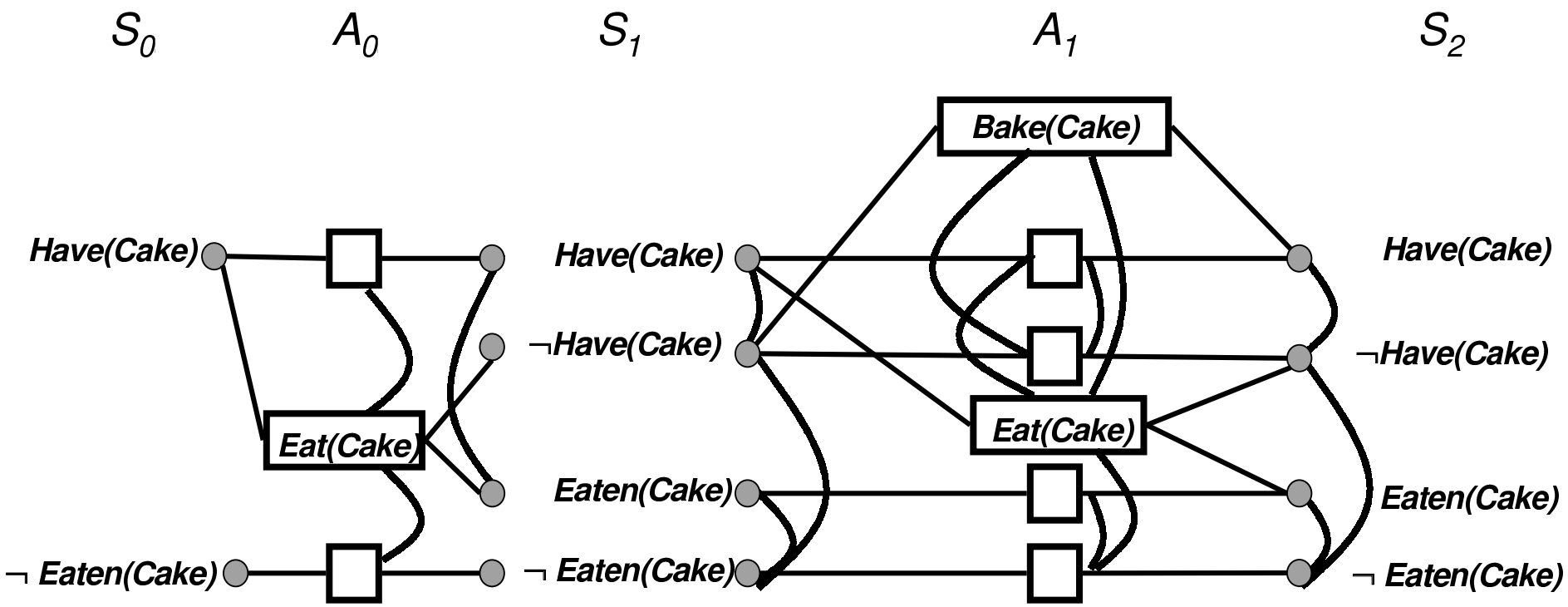
Effect: $\neg$ *Have*(*Cake*) $\wedge$ *Eaten*(*Cake*))

Action (*Bake*(*Cake*)

Precond: $\neg$ *Have*(*Cake*)

Effect: *Have*(Cake))

Grafo de Planejamento



Domínio Jantar Surpresa

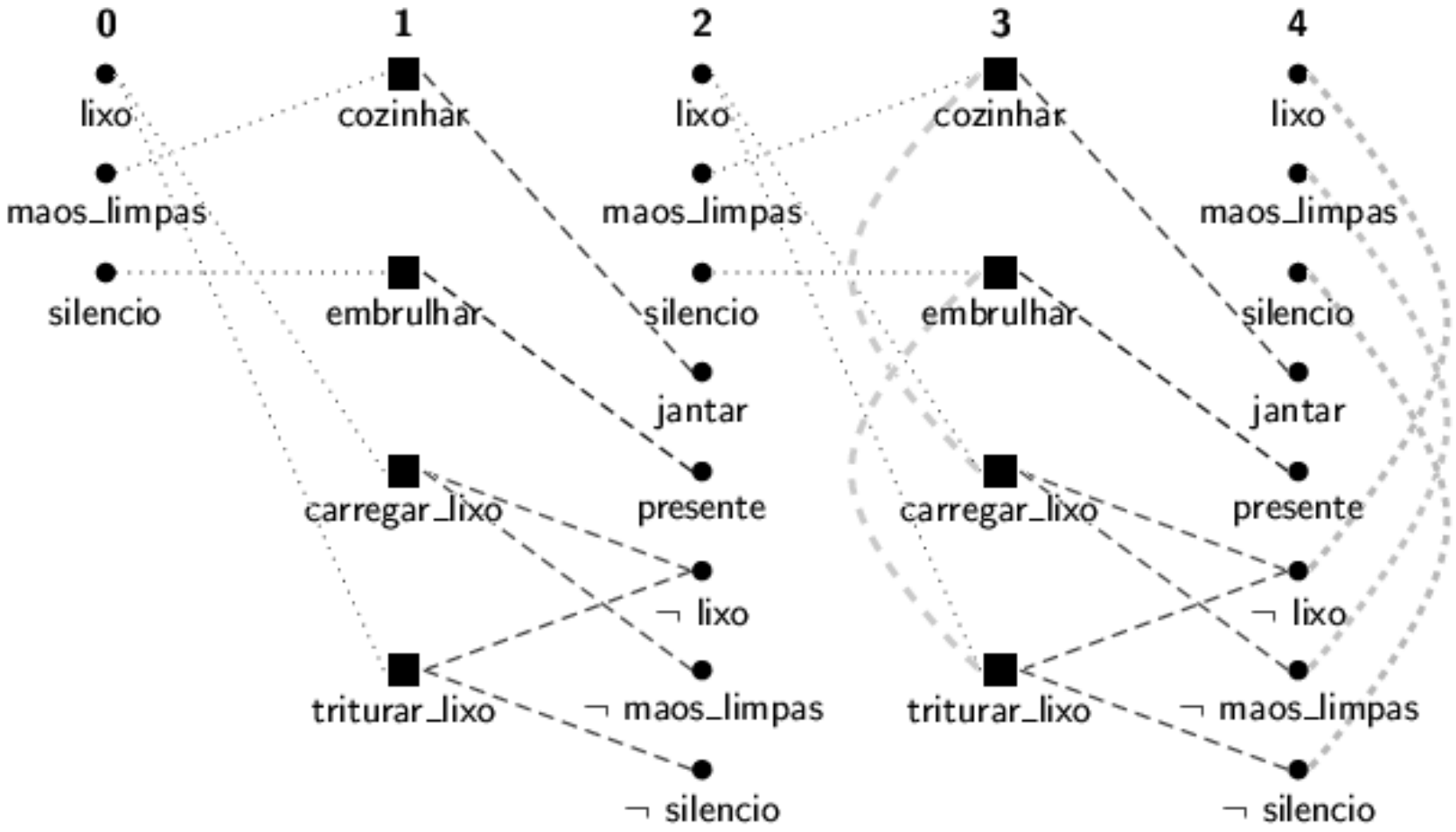
cozinhar :prec (mãos_limpas)
:efeito (jantar)

embrulhar :prec (silêncio)
:efeito (presente)

carregar_lixo :prec (lixo)
:efeito (¬lixo, ¬mãos_limpas)

triturar_lixo :prec (lixo)
:efeito (¬lixo, ¬silêncio)

Grafo de Planejamento



Leliane Nunes de Barros
DCC - IME - USP

Algoritmo Graphplan

- Dois passo que alternam dentro de um laço
  - checa se todos as sub-metas estão no último nível do grafo (cond-1)
    - se isto for verdade então pode existir uma solução no grafo atual e se tenta extrair uma solução.
  - caso a cond-1 não seja satisfeita ou não seja possível extrair uma solução, o grafo deve ser expandido.

Algoritmo Graphplan

function Graphplan(*problem)* **returns** *solution ou failure*

graph ← Initial-Planning-Graph(*problem*)

goals ← Goals[*problem]*

loop do

if *goals* all non-mutex in last level of *graph* **then do**

solution ← Extract-Solution(*graph, goals,* Length(graph))

if *solution* ≠ *failure* **then return** *solution*

else if No-Solution-Possible(*graph*) **then return** *failure*

graph ← Expand-Graph(*graph*, *problem*)

Extração da solução

É preciso usar algumas heurísticas para guiar a escolha das ações durante a busca regressiva

Exemplo: um algoritmo guloso baseado no *level cost* dos literais. Para qualquer conjunto de sub-metas:

1. Selecione primeiro o literal de maior custo
2. Para atingir esse literal, escolha a ação com pré-condições mais "fáceis" (pré-condição com menor **max-level** ou **level sum**)

Condição de Término do Graphplan

- Graphplan pode entrar num laço infinito
  - literais crescem monotonicamente (persistência dos literais)
  - ações crescem monotonicamente (literais da precondição estão sempre presentes)
  - o número de mutex decresce monotonicamente (ações mutex no nível A_i serão mutex nos níveis anteriores em que elas aparecerem)

Blackbox
Graphplan + SATplan

Leliane Nunes de Barros
DCC - IME - USP

Introdução

- **SATplan**
  - métodos para compilar problemas de planejamento em fórmulas proposicionais (CNF) resolvidas por algoritmos SAT sistemáticos e estocásticos.
- **Graphplan**
  - planejador de propósito geral baseado nas idéias usadas em algoritmos de grafos. Dado um problema de planejamento, Graphplan constrói e anota um grafo de planejamento, no qual um plano é um tipo de "fluxo" de valores verdade através do grafo.

Essas duas abordagens envolvem técnicas avançadas de busca e satisfação de restrições que revolucionaram a área em meados dos anos 90.

SATplan

- Mostrou que um provador de teoremas geral pode ser competitivo com alguns dos melhores planejadores da época
  - uso de lógica proposicional
  - convenção de representação de tempo e ações
  - uso de novos algoritmos de raciocínio geral (SAT-solvers)

Graphplan e Satplan

- apresentam desempenho comparável para problemas do Mundo dos Blocos
- SATplan é mais rápido em domínios complexos de logística
- Graphplan é mais rápido em vários outros domínios
- [Khambhampati, 97] [Weld, 1998]

Diferenças e semelhanças

Graphplan: recebe um conjunto de operadores Strips e usa algoritmos específicos para encontrar um plano solução

SATplan: recebe como entrada um conjunto de esquemas de axiomas e usa algoritmos gerais para encontrar um plano solução

Kambhampati: define a classe de *planejadores disjuntivos*

Kautz & Selman: mostram como traduzir um grafo de planejamento em uma CNF==> BlackBox

Semelhanças

Ambas as abordagens trabalham em 2 fases:

(1) criação de uma estrutura proposicional (CNF ou grafo de planejamento)

(2) realizam uma busca (sobre atribuição de variáveis ou em caminhos do grafo) restrita pela estrutura proposicional

A estrutura proposicional corresponde a um tamanho fixo de plano e a busca informa se um plano daquele tamanho existe, caso contrário, a estrutura é espandida

Hipótese (Blackbox)

Graphplan: usa um algoritmo melhor para instanciar a estrutura proposicional

SATplan: usa algoritmos mais poderosos para busca (extração)

Blackbox: resposta ao desafio lançado no IJCAI'97 Subarao Khambhampati: *Como unificar as 2 abordagens?*

SATplan: estrutura proposicional

SATplan: usa um simplificador lógico geral (algoritmo de inferência limitada de tempo polinomial)

Exemplos de regras de inferência de simplificação:

- *Propagação da Cláusula Unitária* (*Unit Propagation*)
- *Purificação* (*Purification*)

SATplan: estrutura proposicional Simplificação

Propagação da Cláusula Unitária

- Se existir uma cláusula *unitária,* existe somente uma atribuição possível para a variável correspondente (restrição unária)
- Exemplo: (x1 + ~x2 + x3)(x2)(x1 + x5)(x2 + x4)
- Faça (x2 = 1):
  - elimine a cláusula em que a variável x2 for positiva
    Ex: (~~x2~~)(x1 + x5)(~~x2 + x4~~)
  - elimine a variável x2 quando for negada Ex: (x1 + ~~~x2~~ + x3)

SATplan: estrutura proposicional Simplificação

Purificação

- Uma variável *pura* sempre aparece na forma positiva (x_i) ou negativa ($\sim x_i$)
- Faça $x_i = 1$ no caso positivo e elimine todas as cláusulas em que ela aparece
- Faça $x_i = 0$ no caso negativo e elimine essa variável de todas as cláusulas em ela aparece

Graphplan: estrutura proposicional
Propagação de Mutex

- A estrutura enquanto um algoritmo de simplificação é aplicado, baseado na computação de mutex (determina que par de ações ou de proposições são mutex)
- A computação de mutex é uma aplicação limitada da "propagação negativa binária"

  dado: {~p v ~q}, {p v ~r}

  infere: {~q v ~r}

Graphplan: estrutura proposicional
Propagação de Mutex

- Cada aplicação da regra permite que se deduza uma cláusula de negação binária (um mutex)
- O algoritmo mutex não é completo (nem todos os mutex que podem ser inferidos logicamente são descobertos) e termina em tempo polinomial

Graphplan: estrutura proposicional

- O conjunto de mutex é usado de 2 maneiras diferentes:
  (1) para cortar nós no grafo durante a instanciação (??)
  (2) para cortar ramos que envolvam ações mutex na árvore de busca

Blackbox

- Combina o melhor das 2 abordagens
- Trabalha em várias fases:

  (1) é gerado um grafo de planejamento de comprimento k e os mutex são computados

  (2) O grafo de planejamento é convertido para uma CNF (usando codificação binária negativa)

  (3) a CNF é simplificada por um algoritmo geral

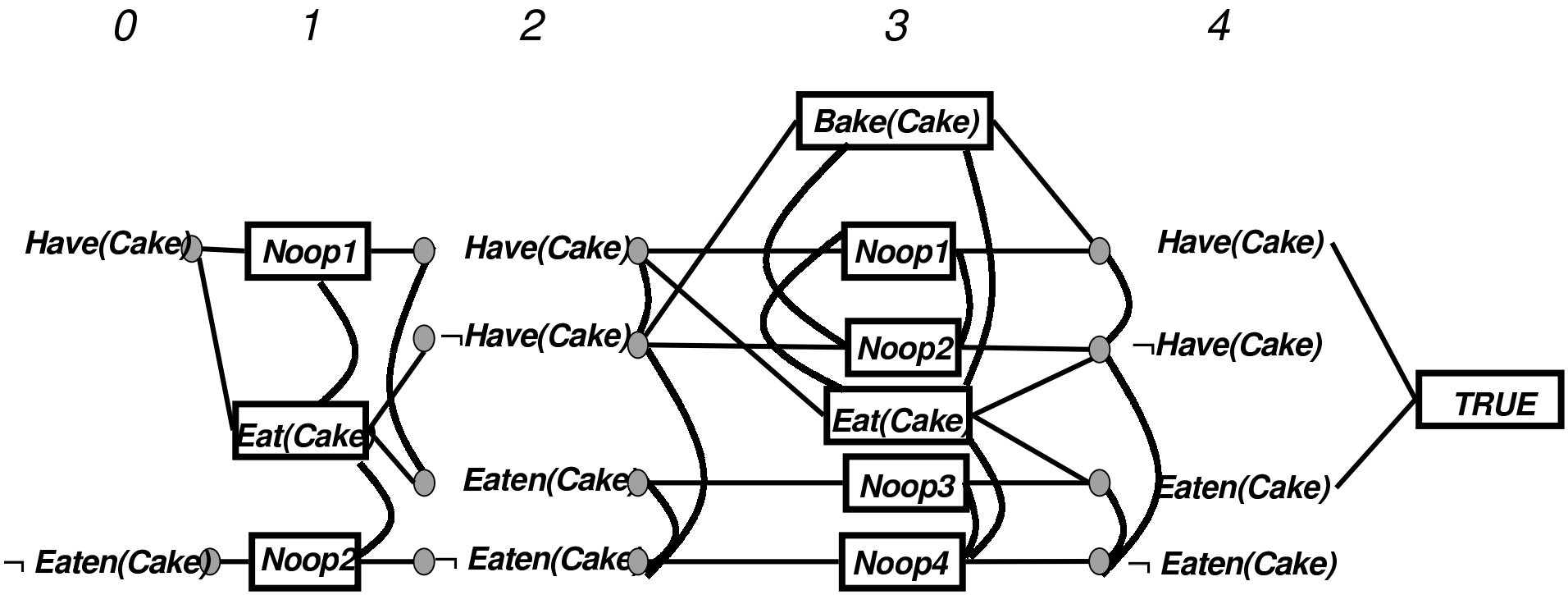
  (4) a CNF é resolvida por qualquer SAT-solver

  (5) se o modelo da CNF é encontrado então ele é convertido para o plano correspondente, caso contrário, k é incrementado e processo se repete

Blackbox

- Tradução do grafo de planejamento para uma CNF:
  - as relações de mutex são traduzidas diretamente em cláusulas binárias negativas ==> gerando fórmulas mais fáceis de serem resolvidas por um SAT-solver.
  - a construção da CNF é feita a partir do último nível

Grafo de Planejamento



Blackbox

- Inicialmente, criamos proposições para as sub-metas do problema
- Sub-metas do problema são vistas como precondições da ação fictícia TRUE
- Para cada sub-meta devemos selecionar a lista de ações que a adiciona.
- No exemplo:
  - Sub-metas de TRUE são ***Eaten_cake(4)*** e ***Have_cake(4)***

Sub-meta ***Eaten_cake(4)***

- Lista de ações que a adiciona:

$$Eat_cake(3) \;\; e \;\; Noop_3(3)$$

- para essas ações são criados símbolos proposicionais para o SAT-solver
- **CLÁUSULA tipo1:**
  - *se a ação TRUE for executada, devo escolher uma ação que satisfaça cada uma de suas precondições*

$$\text{TRUE} \rightarrow Eat_cake(3) \;\vee\; Noop_3(3)$$

- **CLÁUSULA tipo 2:**
  - *somente uma das ações precisa ser executada*

$$\sim Eat_cake(3) \;\vee\; \sim Noop_3(3)$$

Sub-meta ***Have_cake(4)***

- Lista de ações que a adiciona:

$$Bake_cake(3) \;\; e \;\; Noop_1(3)$$

- para essas ações são criados símbolos proposicionais para o SAT-solver
- **CLÁUSULA tipo1:**
  - *se a ação TRUE for executada, devo escolher uma ação que satisfaça cada uma de suas precondições*

$$\text{TRUE} \rightarrow Bake_cake(3) \;\vee\; Noop_1(3)$$

- **CLÁUSULA tipo 2:**
  - *somente uma das ações precisa ser executada*

$$\sim Bake_cake(3) \;\vee\; \sim Noop_1(3)$$

Mutex

- **CLÁUSULA tipo 3:**
  - *todas as relações de MUTEX das ações que possivelmente adicionam as pre-condições de TRUE, devem ser declaradas*

$$\sim Bake_cake(3) \vee \sim Eat_cake(3)$$

$$\sim Noop_1(3) \vee \sim Eat_cake(3)$$

Blackbox

- Selecionamos todas as sub-metas que são precondições das ações possíveis no nível 3:

Bake_cake(3)

~Eat_cake(3)

~Noop_1(3)

~Noop_3(3)

Exemplo

- A lista de precondições de *Eat_cake(3)* é

$$\{Have_cake(2)\}$$

- Criamos proposições para cada elemento dessa lista
- **CLÁUSULA tipo1:**
  - *se a ação TRUE for executada, devo escolher uma ação que satisfaça cada uma de suas precondições*

$$Eat_cake(3) \rightarrow Noop_1(1)$$

- **CLÁUSULA tipo 2:**
  - *somente uma das ações precisa ser executada*

  …neste caso somente Noop_1(1) adiciona Have_cake(2)}

Blackbox

- Inclue:
  - mecanismo para geração do grafo de planejamento
  - 1 SAT-solver de busca local: Walksat
  - 2 SAT-solvers de busca sistemática: satz e rel-sat
    - satz: baseado em *forward-checking*
    - rel-sat: emprega *dependency-directed backtracking*

Blackbox

- Meta-controle:
  - executa o Graphplan por 30 s
  - executa o Walksat por 2 min
  - executa o satz por 5 min
- Blackbox introduziu uma nova técnica de SAT: métodos completos de busca com aleatoriedade

Resultados

- Comparação de desempenho entre Graphplan, SATplan e Blackbox [1]

Resultados

- Re-início aleatório melhora muito o desempenho de algoritmos SAT sistemáticos para problemas de planejamento
  - exemplo: domínio de logística. Backbox: Cutoff de 20 a 200 backtrackings (redução de 1/2 do tempo). SATplan: 16 backtrackings (redução de 1.8 hs para 1.6 min)

Resultados

- Codificação SAT se torna muito grande em domínios sequenciais com muitos operadores
  - ex: mundo dos blocos com um braço (grafo de planejamento deve conter tantas camadas quanto o no. de passos do plano)
  - para n blocos $O(n^2)$ ações e $O(n^4)$ mutex por nível.
  - A tradução de um problema de 15 blocos e de 28 passos de plano solução contém cerca de 2.5 milhões de cláusulas
  - É preciso fazer uma compressão da CNF: adicionar mais conhecimento sobre a estrutura do problema na CNF)

Resultados

- Para problemas muito restritivos em que a extração do plano é o gargalo computacional do Graphplan, a tradução do grafo de planejamento em SAT e uso de um SAT-solver apresenta um desempenho excepcionalmente maior

Propostas de extensão

- Uso de conhecimento específico do domínio como restrições de estado
- Uso de novas técnicas polinomiais de simplificação

Bibliografia

[1] Unifying SAT-based and Graph-based Planning. H. Kautz & Bart Selman. Workshop on Logic-Based Artificial Intelligence, 1999

[2] BLACKBOX: a New Approach to the Application of Theorem Proving to Problem Solving. H. Kautz & Bart Selman. Workshop notes of the Workshop Planning as Combinatorial Search. AIPS, 1998.