Sztuczna inteligencja. Algorytmy i modelowanie świata za pomocą logiki

Paweł Rychlikowski

Instytut Informatyki UWr

13 czerwca 2018

Algorytm DPLL (2)

Definicje

a) **Klauzula jednostkowa** (unit clause) – klauzula zwierająca 1 literał

b) **Czysty literał** – literał, który występuja tylko jako pozytywny, lub tylko jako negatywny (czyli z jedną polaryzacją).

Dwa rodzaje wnioskowania, korzystające z tych pojęć:

a) unit propagation – klazule jednostkowe można spełnić na 1 sposób (spełniając literał), wstawiając wartość logiczną do innych klauzul możemy zrobić nowe klauzule jednostkowe.

b) „Opłaca się” przypisywać czystym literałom wartość true (bo?).

Algorytm DPLL

Algorytm

Funkcja **DPLL**(Φ):

- jeśli Φ zawiera pustą klauzulę zwróć false
- dla każdej klazuli jednostkowej wykonaj unit propagation zmieniając Φ (do nasycenia)
- ustal wartości dla czystych literałów (zmieniając Φ)
- wybierz zmienną x (o nieokreślonej do tej pory wartości)
- zwróć **DPLL**($\Phi \wedge x$) or **DPLL**($\Phi \wedge \neg x$)

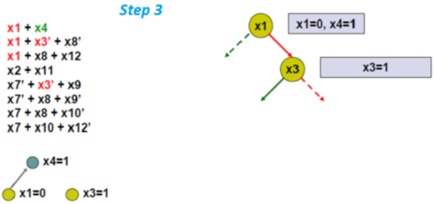
Oczywiście czasem wystarczy sprawdzić tylko jedną część rekurencyjnego wywołania!

Conflict-driven clause learning (CDCL)

W stosunku do DPLL mamy dwie istotne modyfikacje:

1. Po dojściu do sprzeczności możemy dodać nową klauzulę, która podsumowuje przyczynę sprzeczności
2. Przy nawrocie możemy cofnąć się do wcześniejszej zmiennej („praprzyczyny sprzeczności")

Przykład działania CDCL



- Przeanalizujemy zaczerpnięty z Wikipedii przykład działania CDCL.
- Notacja:
  - Mamy literały: **niezwartościowane**, **pozytywne** oraz **negatywne**.
  - Mamy graf implikacji, w którym zapisujemy, jakie konsekwencje powodują nasze wybory
  - Wybory „dowolne” są brudnożółte.

Uwaga

Dla czytelności przykład nie uzględnia obsługi **czystych literałów**!

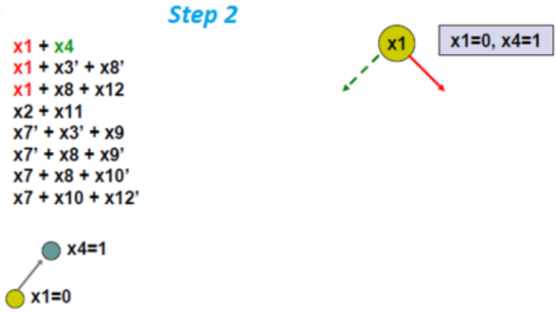
Przykład działania CDCL z Wikipedii

Step 1

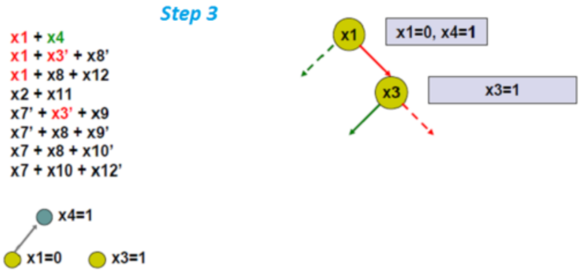
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

x1=0

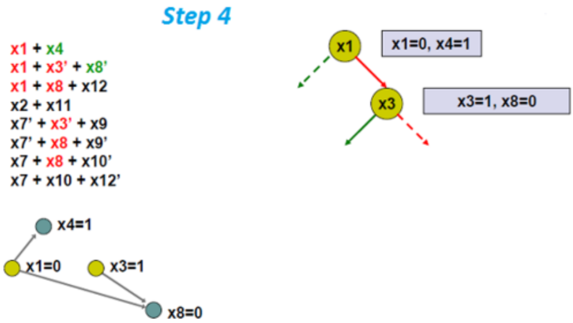
Przykład działania CDCL z Wikipedii



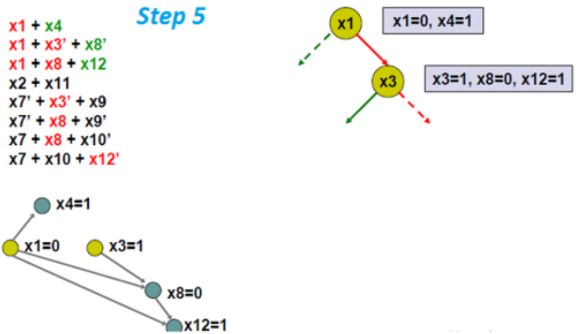
Przykład działania CDCL z Wikipedii



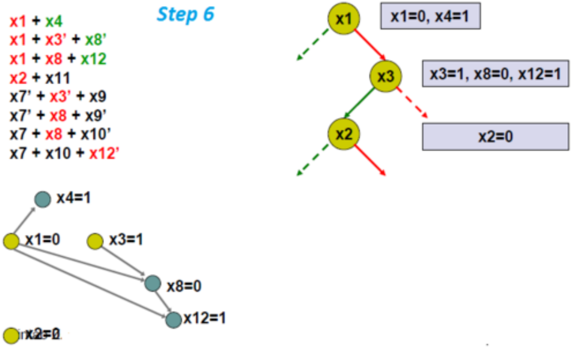
Przykład działania CDCL z Wikipedii



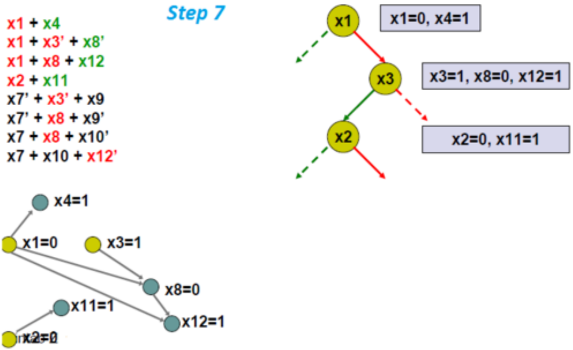
Przykład działania CDCL z Wikipedii



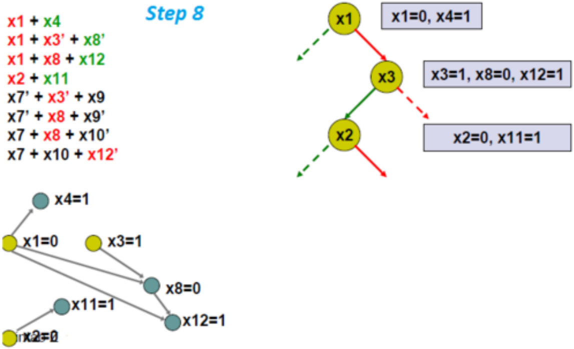
Przykład działania CDCL z Wikipedii



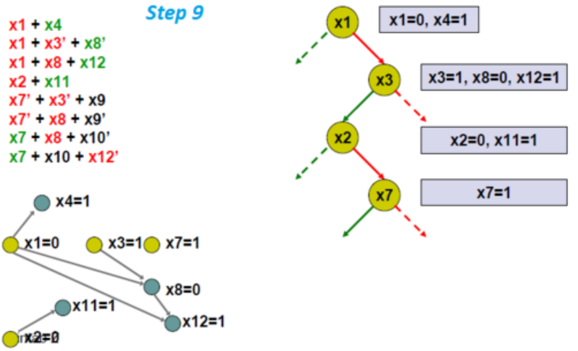
Przykład działania CDCL z Wikipedii



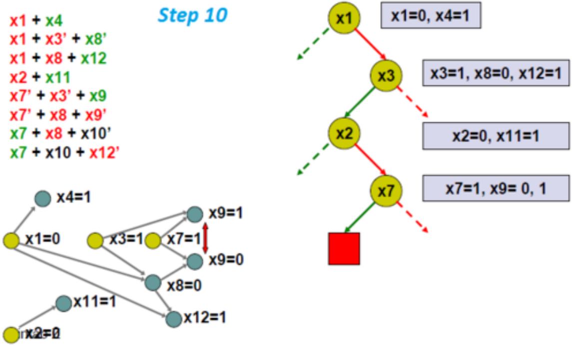
Przykład działania CDCL z Wikipedii



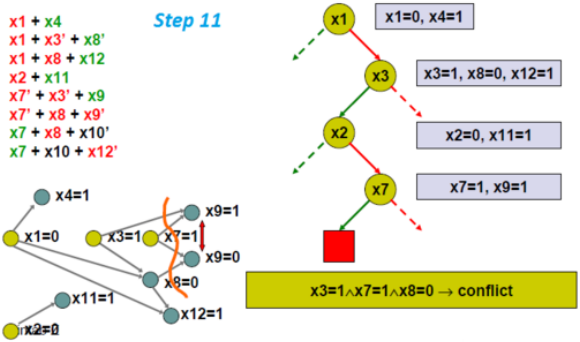
Przykład działania CDCL z Wikipedii



Przykład działania CDCL z Wikipedii



Przykład działania CDCL z Wikipedii



Przykład działania CDCL z Wikipedii

If a implies b, then b' implies a'

Step 12

$x3=1 \wedge x7=1 \wedge x8=0 \rightarrow$ conflict

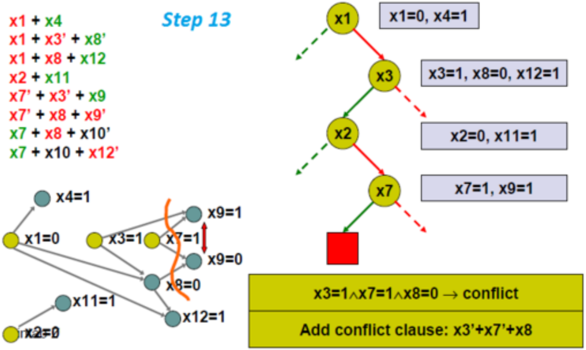
Not conflict $\rightarrow (x3=1 \wedge x7=1 \wedge x8=0)'$

true $\rightarrow (x3=1 \wedge x7=1 \wedge x8=0)'$

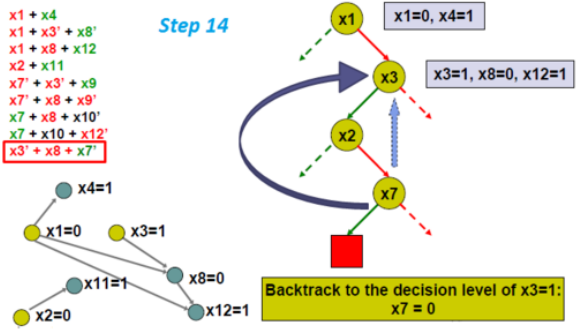
$(x3=1 \wedge x7=1 \wedge x8=0)'$

$\boxed{(x3' + x7' + x8)}$

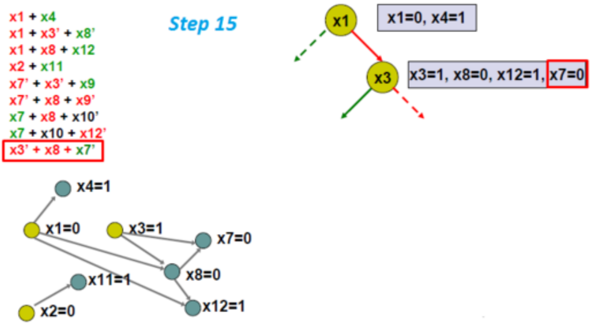
Przykład działania CDCL z Wikipedii



Przykład działania CDCL z Wikipedii



Przykład działania CDCL z Wikipedii



Problem z CDCL

- Trzeba zarządzać „nowymi” klauzulami, monotorować ich przydatność, może kasować...

Uwaga

Obecnie jest to najbardziej efektywna metoda testowania spełnialności (i znajdowania podstawienia).

Alternatywa

- Wypada coś powiedzieć o drugim (również używanym) algorytmie, którym jest ...
  WalkSAT
- (nasz dobry znajomy od obrazków logicznych)

Algorytm WalkSAT

1. Zaczynamy od losowego przypisania zmiennym wartości logicznych.
2. Jak wszystkie klauzule są spełnione (mają co najmniej 1 pozytywny literał), to **koniec**
3. Wybierz losową klauzulę, która jest niespełniona
4. Rzuć monetą (prawdopodobieństwo p):
   a) Orzeł: zmień wartość jednej zmiennej z klauzuli (teraz jest spełniona!)
   b) Reszka: zmień wartość tej zmiennej z klauzuli, która maksymalizuje: różnicę klauzul spełnionych i niespełnionych
5. Po określonej liczbie zmian można zrobić restart, ewentualnie zwrócić stałą **porażka**.

Właściwości WalkSAT

1. **PLUS:** Jak zakończy działanie z sukcesem, to formuła jest spełnialna (i znaleźliśmy podstawienie)
2. **PLUS:** Mamy pełną kontrolę nad czasem działania
3. **MINUS:** Nie możemy mówić o niespełnialności: porażka nic nie oznacza.

Trudność/łatwość wariantów CNF

- Jak już mówiliśmy, nie jest to w pełni satysfakcjonująco rozwiązane.
- Przedstawimy parę spostrzeżeń o formułach losowych

Uwaga

Formuły w CNF możemy łatwo parametryzować (bo mają prostą strukturę)

Definicja

Przez $CNF_k(m, n)$ będziemy rozumieć formułę z k-CNF złożoną z m-klauzul i n-zmiennych.

Losowe CNF

Rozważmy prawdopodobieństwo spełnialności formuły 3-CNF, w zależności od liczby klauzul oraz liczby zmiennych.

- Dużo zmiennych –
- Dużo klauzul –

Losowe CNF

Rozważmy prawdopodobieństwo spełnialności formuły 3-CNF, w zależności od liczby klauzul oraz liczby zmiennych.

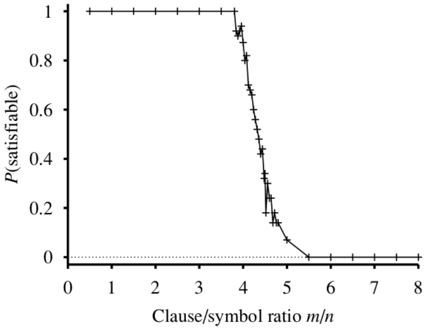
- Dużo zmiennych – łatwo spełnialna
- Dużo klauzul –

Losowe CNF

Rozważmy prawdopodobieństwo spełnialności formuły 3-CNF, w zależności od liczby klauzul oraz liczby zmiennych.

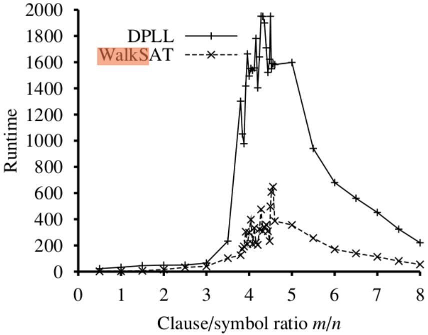
- Dużo zmiennych – łatwo spełnialna
- Dużo klauzul – trudno spełnialna

Losowe CNF. Prawdopodobieństwo spełnienia



- Dużo zmiennych: może wiele z jedną polaryzacją?
- Dużo klauzul – a każda musi być spełniona

Losowe CNF. Czas trwania



Punkt krytyczny w okolicy **m/n = 4.3**

Co teraz?

Wynikanie i wnioskowanie

Wynikanie

Przypomnienie

Formuła definiuje zbiór modeli $\mathcal{M}$, dla których jest ona prawdziwa. Podobnie można mówić o zbiorze modeli dla bazy wiedzy (czyli koniunkcji formuł).

Definicja

Mówimy KB $\models \phi$ wtedy i tylko wtedy, gdy każdy model KB będzie modelem ϕ ($\mathcal{M}(KB) \subseteq \mathcal{M}(\phi)$).

Reguły wnioskowania

Uwaga

Reguły wnioskowania dotyczą syntaktyki, nie semantyki.

Nasłynniejsza reguła wnioskowania

Reguła **modus ponens**: dla dowolnych zmiennych zdaniowych p i q

$$\frac{p,\ p \to q}{q}$$

Kilka faktów o Modus ponens

- Oczywiście jest poprawna (czyli wnioski semantycznie wynikają z przesłanek)
- Można ją uogólnić do większej liczby przesłanek

$$\frac{p_1, \ldots, p_n, \;\; p_1 \wedge \cdots \wedge p_n \rightarrow p_{n+1}}{p_{n+1}}$$

Reguły wnioskowania

Ogólna postać reguły wnioskowania jest następująca:

$$\frac{f_1, \ldots, f_n}{g}$$

Algorytm wnioskowania

Wnioskowanie w przód (forward inference)

Powtarzaj, aż do momentu, gdy nie da się zmienić Bazy wiedzy:

- Wybierz $\{f_1, \ldots, f_k\} \subseteq \mathcal{KB}$
- Jeżeli istnieje reguła:

$$\frac{f_1, \ldots, f_n}{g}$$

  dodaj g do Bazy wiedzy

Definicja

Jeżeli powyższy algorytm dodaje f w którymś momencie do bazy wiedzy, wówczas piszemy $KB \vdash f$

2 proste uwagi o wnioskowaniu

1. Wnioskowanie w tył: Zaczynamy od tego, co chcemy udowodnić (od naszego celu).
2. Możemy myśleć o dowodzeniu twierdzeń jako o zadaniu przeszukiwania.
   (przestrzenią stanów są zbiory aksjomatów i dowiedzionych formuł, celem – zbiór zawierający docelowe twierdzenie)

Dygresja. Klauzule Hornowskie

Definicja

Klauzula Hornowska to taka klauzula, która ma **co najwyżej** jeden literał pozytywny.

Przykłady

- p_1 (fakty)
- $\neg p_2$ (zaprzeczenia faktów)
- $\neg p_2 \vee p_3$ (czyli $p_2 \to p_3$)
- $\neg q_1 \vee \ldots \neg q_n \vee q_{n+1}$ (czyli $q_1 \wedge \cdots \wedge q_1 \to q_{n+1}$)

Uwaga

Klauzule Hornowskie mają duże znaczenie w Programowaniu logicznym (programy w Prologu składają się z klauzul hornowskich).

Wnioskowanie w przód dla klazul hornowskich. Modus ponens

- Wnioskujemy tylko pozytywne fakty.
- Możemy zauważyć sprzeczność, jeżeli wydedukujemy p, a w bazie wiedzy mieliśmy $\neg p$

Operacje na Bazie wiedzy

Operacja Tell(KB, ϕ)

Dodaje formułę ϕ do bazy wiedzy (proste dodanie do zbioru)

Operacja Ask(KB, ϕ)

Sprawdza, czy KB $\vdash \phi$.

Czesto realizujemy operację **Ask** sprawdzając, czy $KB \wedge \neg\phi$ jest spełnialne/sprzeczne.

Wynikanie i dowodzenie

Definicja 1

Logika jest poprawna, jeżeli $M \vdash \phi$ implikuje $M \models \phi$

Definicja 2

Logika jest zupełna, jeżeli $M \models \phi$ implikuje $M \vdash \phi$

Uwaga

Poprawność jest konieczna, zupełność – porządana.

Zupełność i poprawność

- The truth, the whole truth, and nothing but the truth.

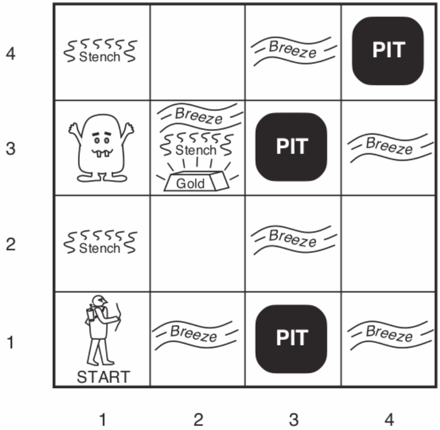
Przykład. Zupełność (?) modus ponens

Modus ponens **nie** jest zupełny

Przykład

$\mathcal{KB} = \{\text{deszcz}, \text{deszcz} \vee \text{śnieg} \to \text{mokry}\}$
Mokry jest prawdziwe, ale niedowodliwe.

Modelowanie świata za pomocą logiki



- Wumpus śmierdzi, złoto błyszczy, w szybie są przeciągi.
- Poruszamy się o jedną kratkę w 4 kierunkach.
- Mamy jedną strzałę (strzela po liniach prostych).
- **Znamy mechanikę, ale nie znamy konkretnej edycji świata, odbieramy go za pomocą bodźców.**

Modelowanie świata za pomocą logiki zdaniowej

Uwaga

Musimy opisać świat za pomocą skończonej liczby bitów

Przykładowe zmienne:

1. Położenie dziur, wumpusa, złota: $P_{1,2}$, $W_{4,4}$, $G_{3,2}$
2. Położenie miejsc „z bodźcami": $S_{2,2}$, $B_{1,2}$
3. Położenie agenta: $L^t_{3,3}$ (konieczne uwzględnienie czasu)
4. Wrażenia agenta: Breeze^t, Stench^t
5. Stan agenta w chwili t, akcja agenta w chwili t, itd

Przykładowe fragmenty modelu

- Jeżeli gdzieś jest przeciąg, to w okolicy jest dziura: $B_{1,1} \leftrightarrow P_{2,1} \vee P_{1,2}$
- Jest (co najmniej) jeden Wumpus: $W_{1,1} \vee W_{1,2} \vee \cdots \vee W_{4,4}$
- Jest co najwyżej 1 Wumpus: (w każdych dwóch W co najmniej 1 fałszywy)
- Powiązanie wrażeń agenta ze światem: $L^t_{x,y} \rightarrow (\text{Breeze}^t \leftrightarrow B_{x,y})$

Reguły ruchu

Uwaga

Potrzebujemy dla każdej akcji agenta opisać co się zmieni, a co nie zmieni w świecie.

Przykłady:

- $L_{1,1}^{t} \wedge \mathrm{FacingEast}^{t} \wedge \mathrm{Forward}^{t} \rightarrow (L_{2,1}^{t+1} \wedge \neg L_{1,1}^{t+1})$
- $\mathrm{Forward}^{t} \rightarrow (\mathrm{HaveArrow}^{t} \leftrightarrow \mathrm{HaveArrow}^{t+1})$
- itd

Pamiętamy, że te reguły trzeba powtórzyć dla wszystkich lokacji (i dla różnych czasów, ale o tym za chwilę)

Hybrydowy agent w świecie Wumpusa

- Wykorzystuje procedurę szukania drogi (poruszając się po polach bezpiecznych)
- Gromadzi wiedzę o świecie:
  - Zaobserwowane bodźce (i wnioski z nich płynące)
  - „Rozwijane” zdania o mechanice świata (dla momentu t)
- Zarządza **planem akcji**.

Wumpus Agent (1)

function HYBRID-WUMPUS-AGENT(*percept*) **returns** an *action*
  inputs: *percept*, a list, [*stench*,*breeze*,*glitter*,*bump*,*scream*]
  persistent: KB, a knowledge base, initially the atemporal "wumpus physics"
    t, a counter, initially 0, indicating time
    plan, an action sequence, initially empty

  TELL(KB, MAKE-PERCEPT-SENTENCE(*percept*, t))
  TELL the KB the temporal "physics" sentences for time t
  $safe \leftarrow \{[x, y] : \text{ASK}(KB, OK^t_{x,y}) = true\}$
  if $\text{ASK}(KB, Glitter^t) = true$ **then**
    $plan \leftarrow [Grab] + \text{PLAN-ROUTE}(current, \{[1,1]\}, safe) + [Climb]$
  if *plan* is empty **then**
    $unvisited \leftarrow \{[x, y] : \text{ASK}(KB, L^{t'}_{x,y}) = false \text{ for all } t' \leq t\}$
    $plan \leftarrow \text{PLAN-ROUTE}(current, unvisited \cap safe, safe)$

Wumpus Agent (2)

if $plan$ is empty and $\textsc{Ask}(KB, HaveArrow^t) = true$ **then**
  $possible\_wumpus \leftarrow \{[x,y] : \textsc{Ask}(KB, \neg W_{x,y}) = false\}$
  $plan \leftarrow \textsc{Plan-Shot}(current, possible\_wumpus, safe)$
if $plan$ is empty **then** // no choice but to take a risk
  $not\_unsafe \leftarrow \{[x,y] : \textsc{Ask}(KB, \neg OK^t_{x,y}) = false\}$
  $plan \leftarrow \textsc{Plan-Route}(current, unvisited \cap not\_unsafe, safe)$
if $plan$ is empty **then**
  $plan \leftarrow \textsc{Plan-Route}(current, \{[1,1]\}, safe) + [Climb]$
$action \leftarrow \textsc{Pop}(plan)$
$\textsc{Tell}(KB, \textsc{Make-Action-Sentence}(action, t))$
$t \leftarrow t + 1$
return $action$

Planowanie w logice zdaniowej

Braki logiki zdaniowej

Podstawowy brak: nie ma kwantyfikatorów, czyli pewne ogólne prawdy musimy wyrażać jako skończone alternatywy/koniunkcje.

Przykłady

- Każdy student jest pilny
- Pilni studenci zdają egzaminy, na które sa zapisani.
- Przynajmniej jedna osoba dostanie piątkę z AI

Jeżeli mówimy o skończonej liczbie obiektów, możemy traktować kwantyfikatory jako skróty dla koniunkcji ($\forall$) lub alternatywy ($\exists$)

Konwersja do CNF dla logiki pierwszego rzędu

Definicja

Klauzula (w logice 1-go rzędu) jest formułą:

$$\forall x_1 \forall x_2 \ldots \forall x_n \; A_1 \vee A_k$$

A_i – formuły atomowe.

Pytanie

Jak sobie radzić z kwantyfikatorami podczas konwersji do CNF?

Inne logiki