Logika

Paweł Rychlikowski

Instytut Informatyki UWr

29 maja 2021

Paradygmaty modelowania świata

1. Bazujące na **stanach**: przeszukiwanie, MDP, gry
2. Bazujące na **zmiennych**: CSP, sieci Bayesowskie (jeszcze przed nami)
3. Bazujące na **logice**: logika zdaniowa, logiki modalne, logika 1-go rzędu

Zajmiemy się teraz logiką. Zaczniemy od **logiki zdaniowej**.

Logika zdaniowa

- zmienne zdaniowe (przyjmują wartości 0/1)
- spójniki: $\vee$, $\wedge$, $\rightarrow$, $\leftrightarrow$, $\neg$

Przykłady

- pada $\wedge \neg$ mam-parasol $\rightarrow$ jestem-mokry $\vee$ mam-kurtkę
- $\neg\ (p \vee q) \leftrightarrow (\neg\ p \wedge \neg\ q)$

Model (logika zdaniowa)

- **Modelem** w logice zdaniowej jest przypisanie zmiennym wartości logicznych.
  - Ogólnie o modelu myślimy jako o naszej wizji świata.
- **Interpretacją** formuły przy zadanym modelu jest zdefiniowana rekurencyjnie wartość formuły:
  - $I(a, w) = w(a)$, jeżeli a jest zmienną
  - $I(f_1 \vee f_2, w) = I(f_1, w) \vee I(f_2, w)$
  - (...) – inne, podobne reguły
- Składnia (syntax) vs semantyka

Modele i formuły

Definicja

Formuła f jest **spełnialna**, czyli ma model, jeżeli istnieje takie w, ze $I(f, w) = 1$.

Uwaga

Takich przypisań jest skończenie wiele, stąd mamy prosty (wykładniczy) algorytm sprawdzania, czy formuła ma model (jest spełnialna)

Baza wiedzy

- Formuły to zdania opisujące świat.
- Naturalne jest myślenie o zbiorze takich formuł (do którego możemy dodawać nowe fakty).
- Taki zbiór często nazywamy **bazą wiedzy**.

Koniunkcyjna postać normalna

Definicje

1. literał - zmienna albo $\neg$ zmienna
2. klauzula - $l_1 \vee \cdots \vee l_n$ (gdzie l_i to literał)
3. formuła w **CNF** - $c_1 \wedge \cdots \wedge c_n$, gdzie c_i jest klauzulą

Dlaczego CNF jest fajna?

- Każdą formułę można przekształcić do CNF (czasem płacąc wykładniczym wzrostem jej długości, ćwiczenia)
- Koniunkcja klauzul = zbiór klauzul = baza wiedzy
- Jak mamy zbiór formuł (bazę wiedzy, niekoniecznie w CNF) to wykładniczość dotyczy pojedynczej formuły, a nie całej bazy.

CSP a spełnialność

- Sprawdzanie spełnialności formuły boolowskiej jest zadaniem rozwiązywania więzów (baza wiedzy = zbiór więzów)
- Nie dziwi zatem, że podstawowe algorytmy (stosowane w praktyce) są dość podobne (backtracking + propagacja).

Uwaga

Współczesne SAT-solvery radzą sobie z milionami klauzul i setkami tysięcy zmiennych

A NP-zupełność?

- Oczywiście problem CNF-SAT (spełnialności formuły w CNF) jest NP-zupełny.
- Nie spodziewamy się istnienia algorytmu wielomianowego (znane algorytmy mają pesymistyczny czas wykładniczy).

Pytanie

Dlaczego SAT-Solvery działają dobrze?

Pytanie jest trudne, i tak do końca nie ma odpowiedzi. Jedyne, co można powiedzieć, że widocznie znaczna część w praktyce spotykanych formuł jest w jakimś sensie **łatwa**.

Algorytm DPLL

- Algrytm Davisa–Putnama–Logemanna–Lovelanda (**DPLL**) jest algorytmem znajdującym spełniające podstawienie dla formuły CNF.
- Jest **zupełny** – tzn. zawsze kończy się z prawidłowym wynikiem, może działać długo

Uwaga

Stanowi bazę współczesnych SAT-Solverów (z jednym usprawnieniem, o którym jeszcze powiemy).

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).

$$x_1 \wedge \neg x_2 \wedge (x_3 \vee x_4 \vee x_5) \wedge (\neg x_3 \vee \neg x_4 \vee x_5)$$

Algorytm DPLL (2)

Jakie wnioskowanie można przeprowadzić korzystając 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!

O uczeniu klauzul

- Zauważmy, że jak w algorytmie DPLL osiągnęliśmy sprzeczność, to można myśleć, że „udowodniliśmy" twierdzenie (przy założeniu analizowanej formuły Φ):

$$(l_1 \wedge l_2 \wedge \ldots \wedge l_n) \rightarrow \text{Fałsz}$$

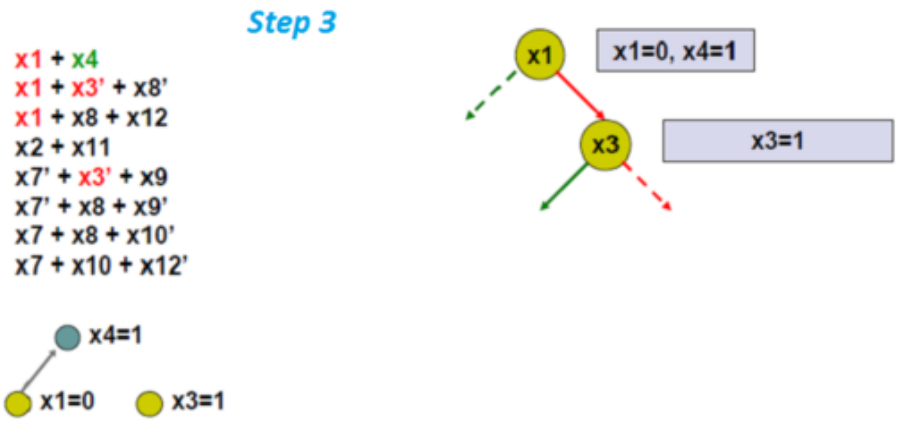
  gdzie l_i jest literałem użytym w i-tym momencie algorytmu (w propagacji lub w backtrackingu)
- Możemy zatem uznać, że udowodniliśmy twierdzenie: ze zbioru klauzul wynika $\neg l_1 \vee \cdots \vee \neg l_n$
- Trochę długa formuła (i nieużyteczna w propagacji), algorytm próbuje zatem znaleźć możliwie krótki ciąg literałów o ustalonych wartościach, który sam z siebie implikuje Fałsz (i zapamiętać go na przyszłość)

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**!

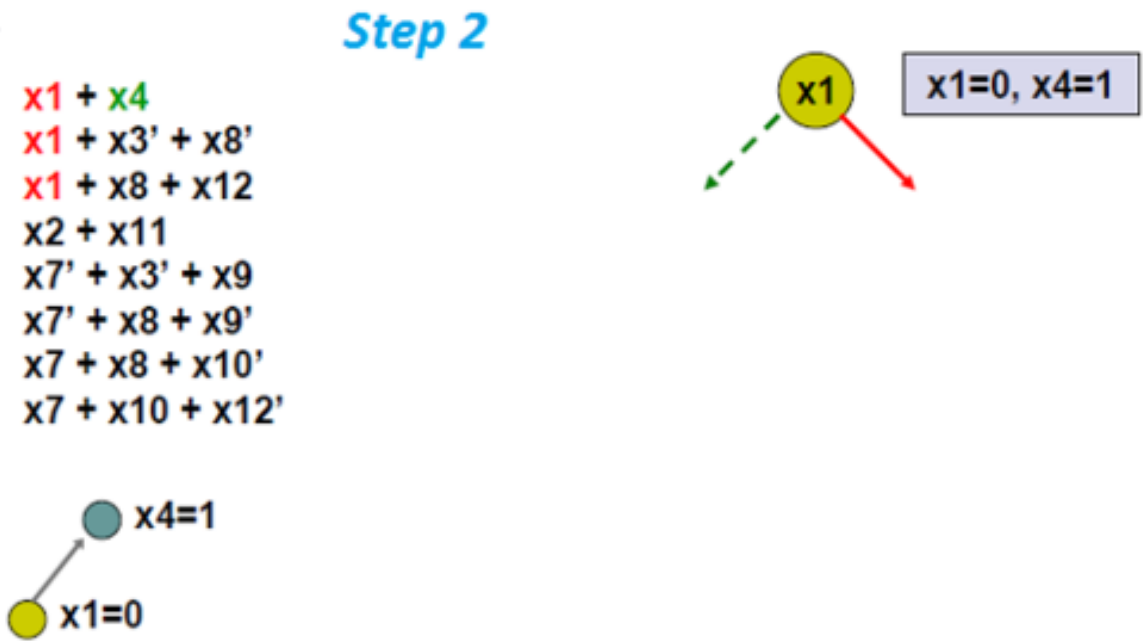
Przykład działania CDCL z Wikipedii

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

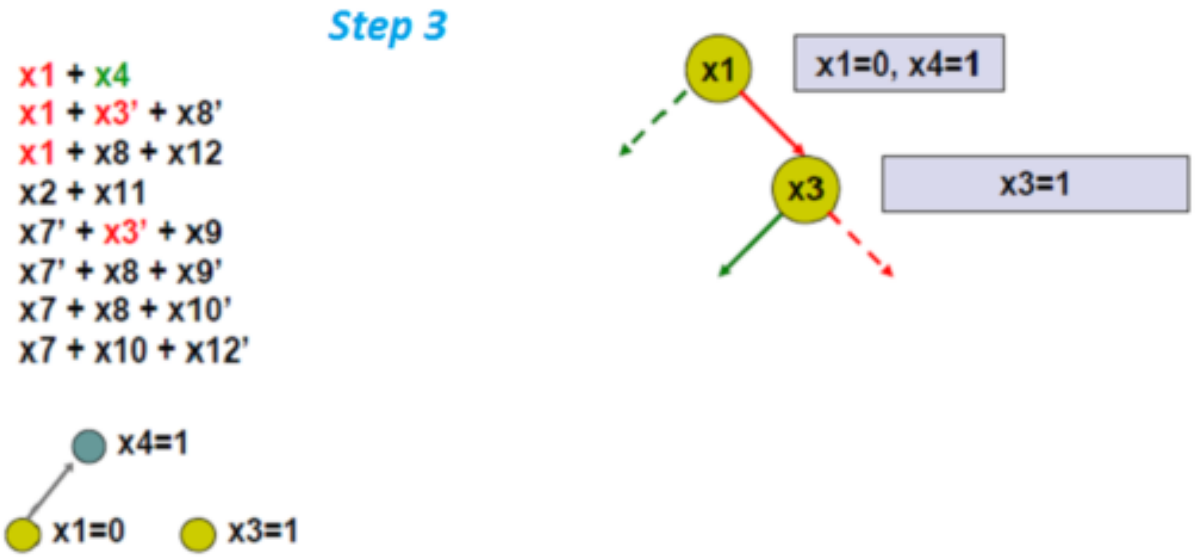
Step 1

x1=0

Przykład działania CDCL z Wikipedii



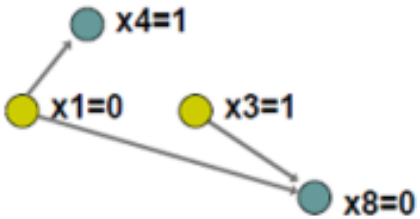
Przykład działania CDCL z Wikipedii

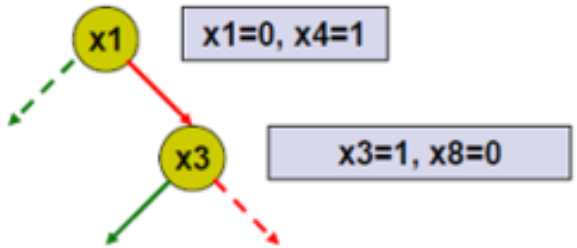


Przykład działania CDCL z Wikipedii

Step 4

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

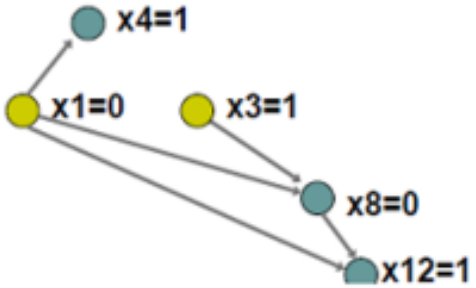


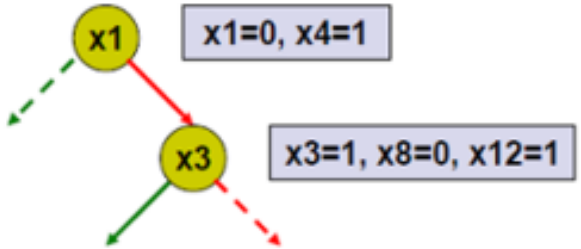


Przykład działania CDCL z Wikipedii

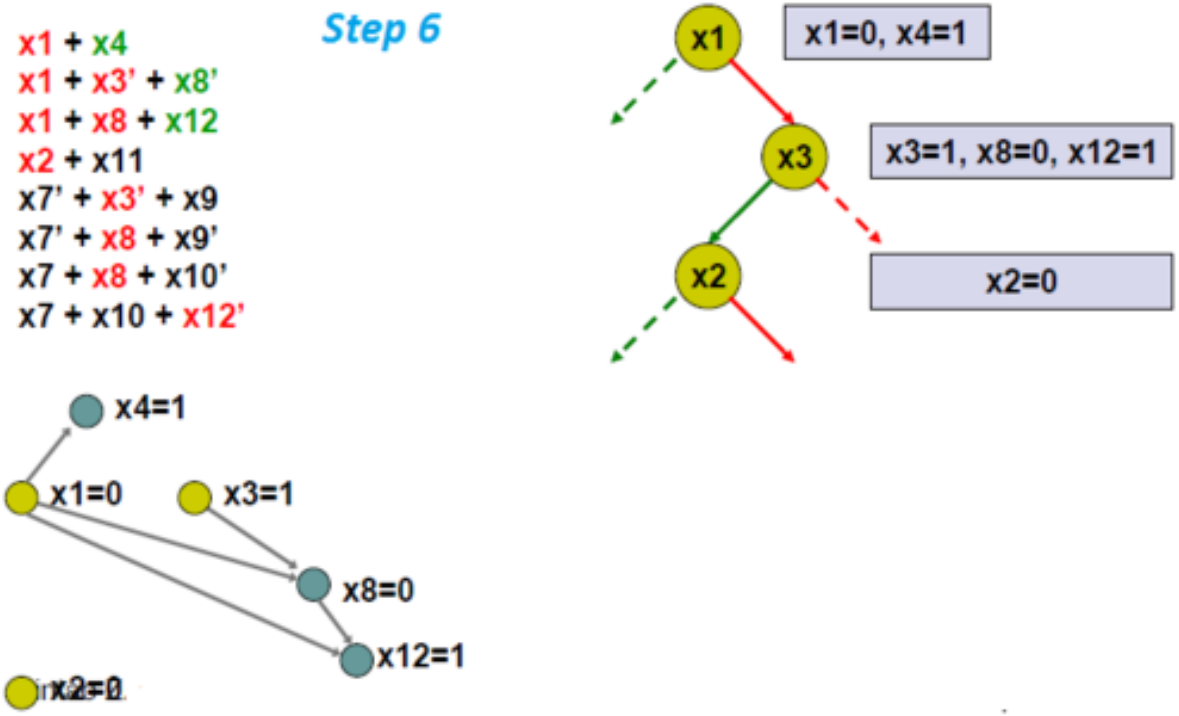
Step 5

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





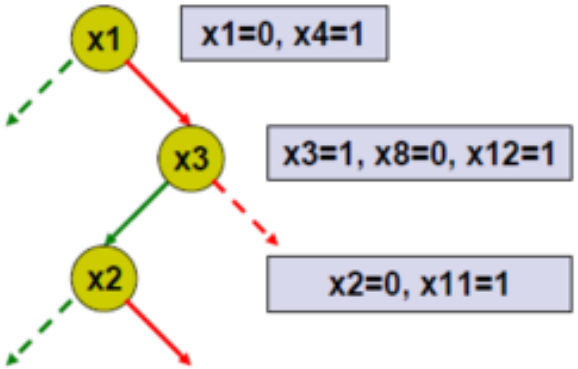
Przykład działania CDCL z Wikipedii

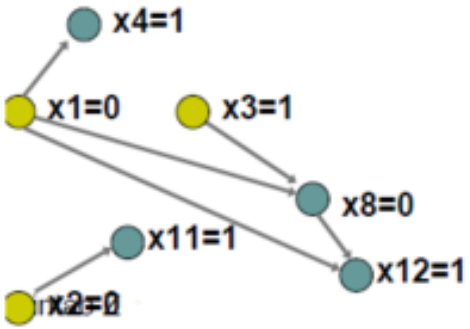


Przykład działania CDCL z Wikipedii

Step 7

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

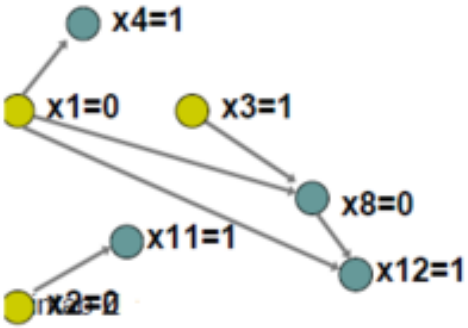


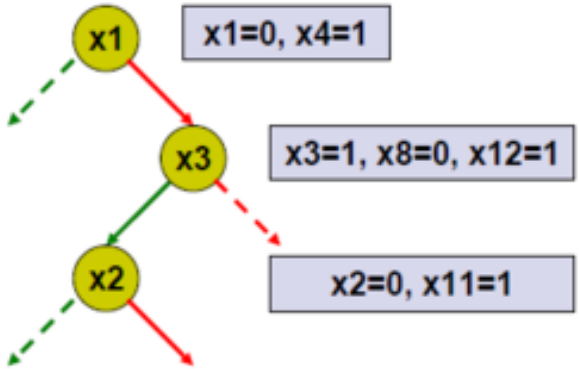


Przykład działania CDCL z Wikipedii

Step 8

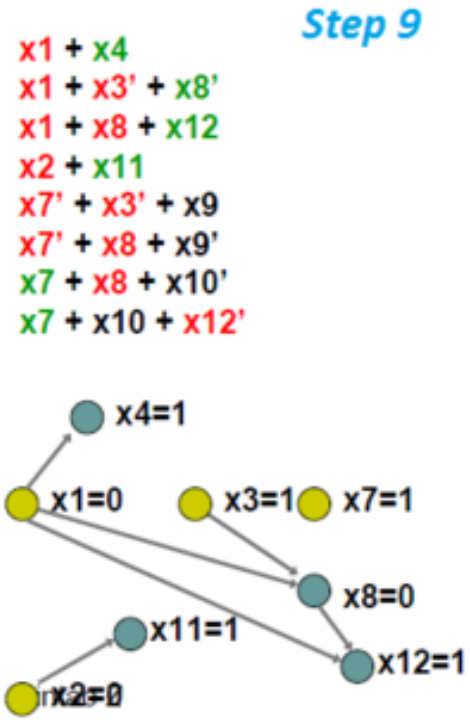
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

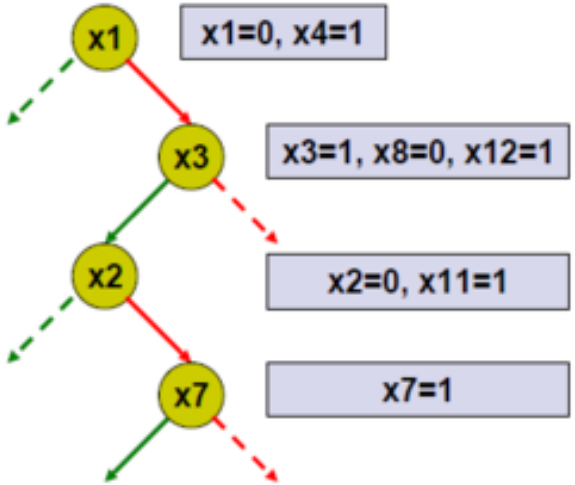




Koniec części I

Przykład działania CDCL z Wikipedii

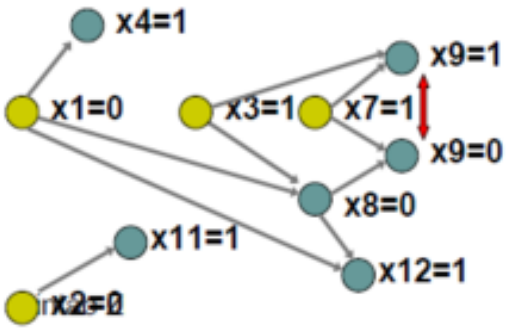


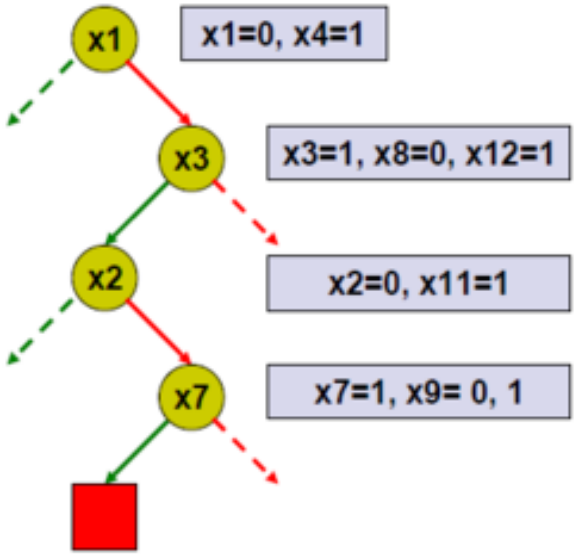


Przykład działania CDCL z Wikipedii

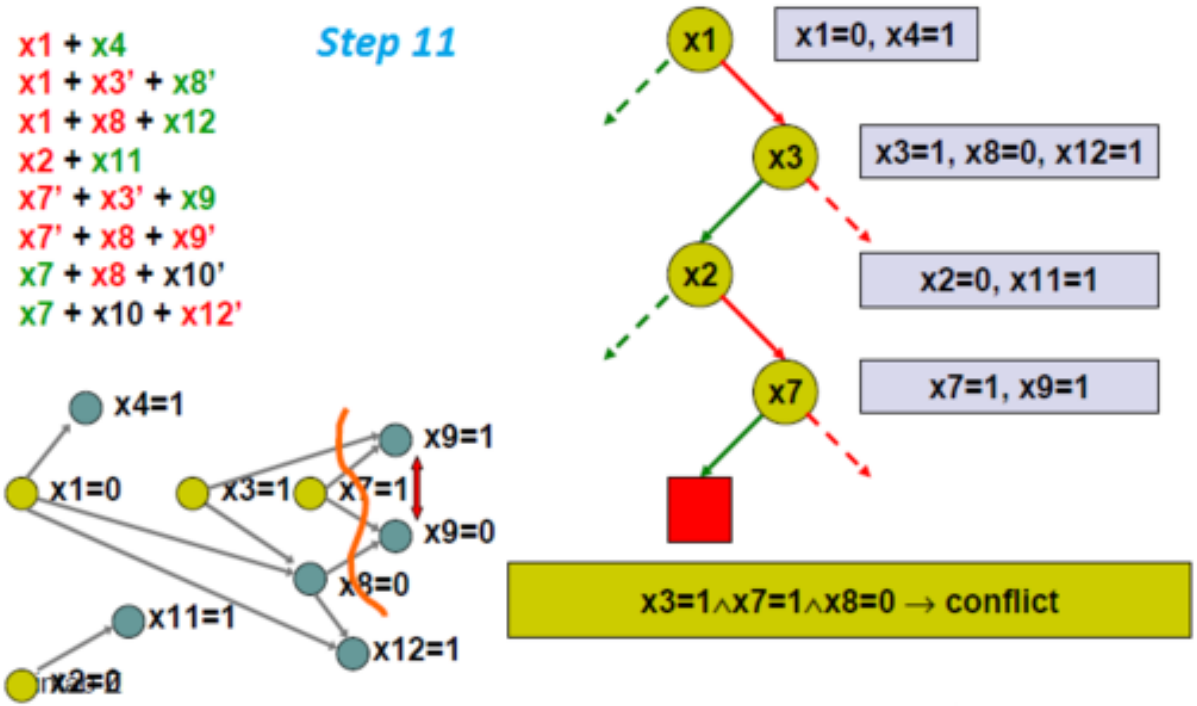
x1 + x4
x1 + x3' + x8'
x1 + x8 + x12
x2 + x11
x7' + x3' + x9
x7' + x8 + x9'
x7 + x8 + x10'
x7 + x10 + x12'

Step 10





Przykład działania CDCL z Wikipedii



Przykład działania CDCL z Wikipedii

If a implies b, then b' implies a'

Step 12

$$x3=1 \wedge x7=1 \wedge x8=0 \rightarrow \text{conflict}$$

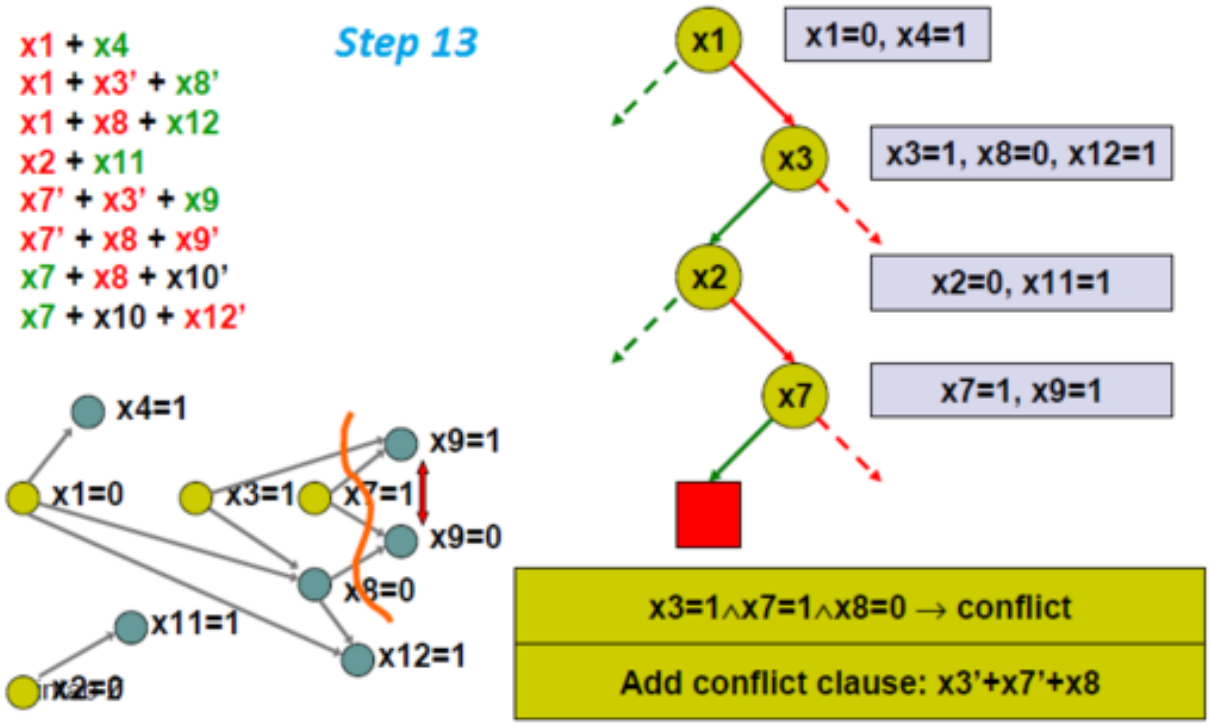
$$\text{Not conflict} \rightarrow (x3=1 \wedge x7=1 \wedge x8=0)'$$

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

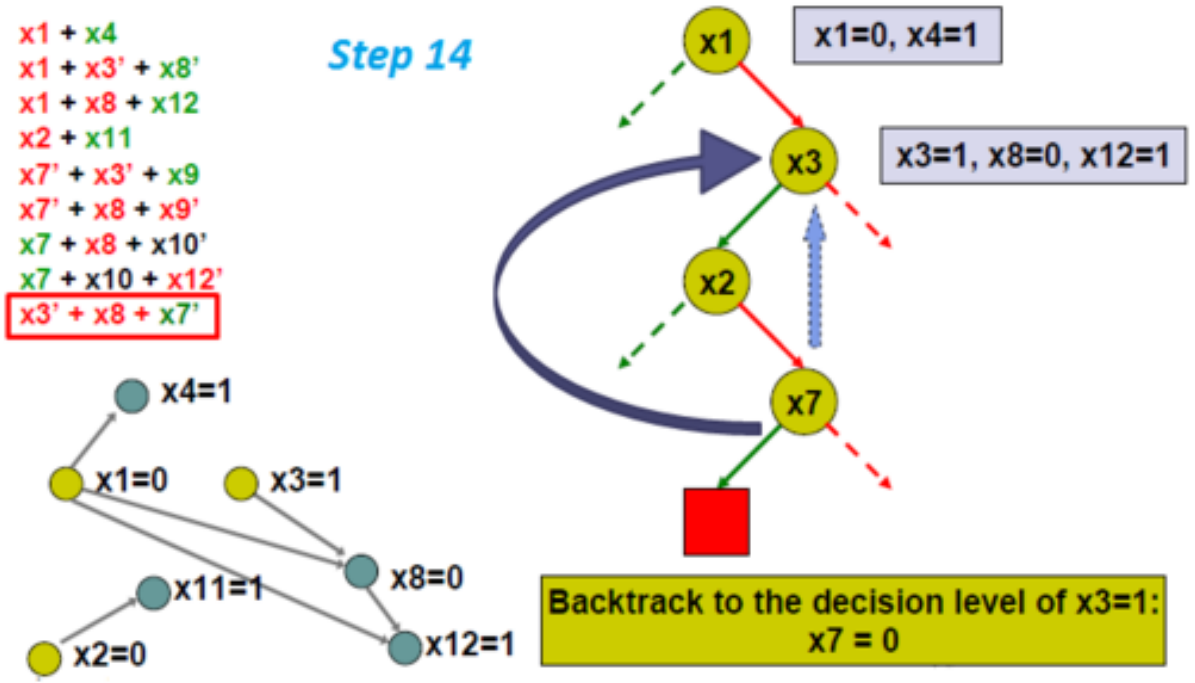
$$(x3=1 \wedge x7=1 \wedge x8=0)'$$

$$(x3' + x7' + x8)$$

Przykład działania CDCL z Wikipedii



Przykład działania CDCL z Wikipedii



Problem z CDCL

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

Uwaga

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

Alternatywa dla DPLL

- Wypada coś powiedzieć o drugim (również używanym) algorytmie, którym jest ...
  WalkSAT

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. Mają one prostą strukturę, daną przez:

- Liczbę różnych zmiennych
- Liczbę klauzul
- Maksymalną wielkość klauzuli

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.

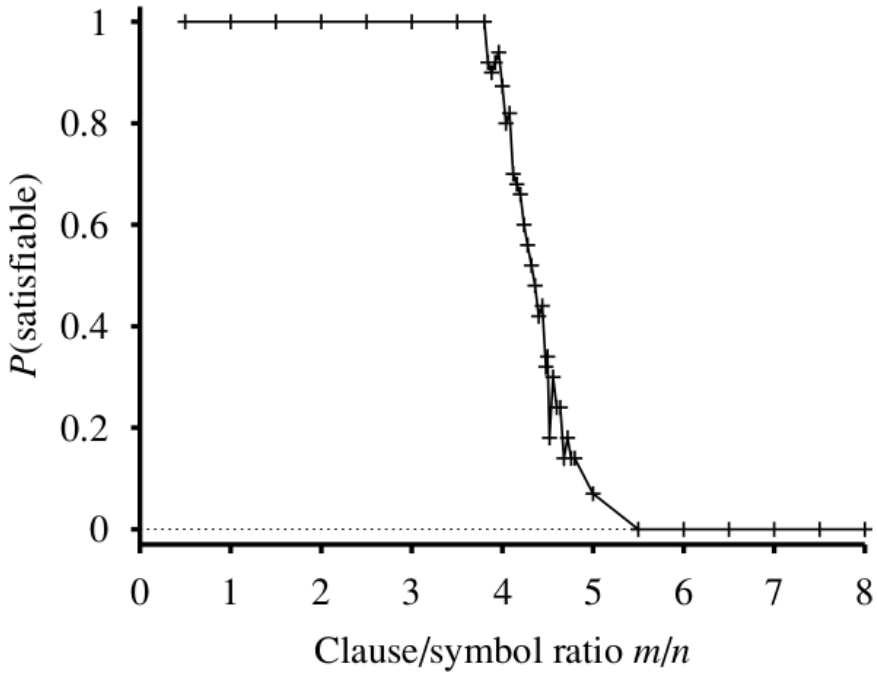
- Dużo zmiennych – łatwo spełnialna
- Dużo klauzul –

Losowe CNF

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

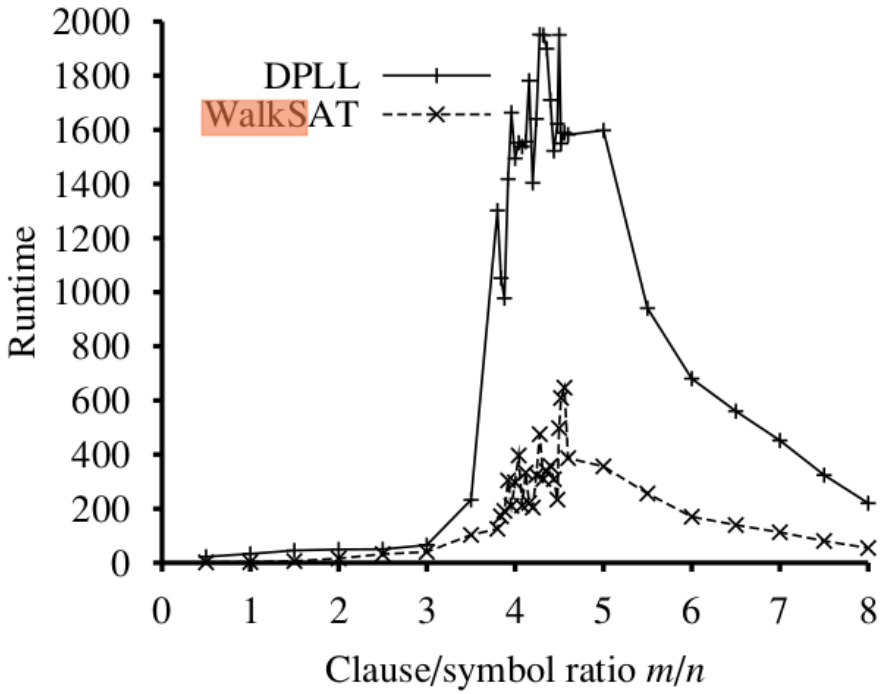
- Dużo zmiennych – łatwo spełnialna
- Dużo klauzul – trudno spełnialna

Losowe CNF. Prawdopodobieństwo spełnienia



- Dużo zmiennych: może wiele z jedną polaryzacją?
- Dużo zmiennych: może dłuższe klauzule (a do spełnienia wystarczy 1 literał)
- Dużo klauzul – a każda musi być spełniona

Losowe CNF. Czas trwania



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

Wykorzystanie logiki zdaniowej

- Zadania z więzami realizowane za pomocą logiki zdaniowej
  Przykład: obrazki **logiczne**
- Algorytmy planowania
  Przeszukiwanie specjalnej przestrzeni stanów, w której dozwolone ruchy opisane są formułami
- Agenty hybrydowe (przeplatanie wnioskowania z innymi metodami)

Obrazki logiczne jako CNF-SAT

Przykładowa reprezentacja

- Mamy zmienne (binarne) odpowiadające polom,
- Mamy zmienne typu: w wierszu 5 jest układ 00111001100

Co dalej?

Formuły

- W każdym wierszu (bądź kolumnie) jest jeden ze zgodnych ze specyfikacją układów (długa alternatywa)
- Formuły „tłumaczące" zmienne wierszowe na zmienne związane z polami:

$$W_{01101110} \rightarrow \neg X_0 \wedge X_1 \wedge X_2 \wedge \neg X_3 \wedge X_4 \wedge X_5 \wedge X_6$$

(to jest skrót dla n formuł typu $W_{01101110} \rightarrow L_i$)

Opisywanie akcji za pomocą formuł

Uwaga

Mamy czas, czyli $t \in 0, \ldots, T-1$. Używamy zmiennych mówiących o stanie świata w momencie t i o akcji w momencie t. Opisujemy mechanikę świata językiem logiki.

- Określamy zmienne prawdziwe w czasie 0,

$$\text{stoję-przed-sklepem}_0 \wedge \text{pusta-torba}_0 \wedge \text{pełen-portfel}_0 \wedge \ldots$$

- Określamy cel (czyli co chcemy uzyskać w czasie T) – czas T musimy zgadnąć, czyli inaczej mówiąc sprawdzać kolejne wartości, aż do skutku.

$$\text{stoję-przed-sklepem}_T \wedge \neg\text{pusta-torba}_T \wedge \neg\text{pełen-portfel}_T$$

Opisywanie akcji za pomocą formuł

Uwaga techniczna

Formuły zapisujemy często w bogatszym języku, zakładając, że system sam je przekształci do CNF-u.

- Warunki wstępne i końcowe poleceń

$$\text{strzelam}_t \rightarrow \text{mam-łuk}_t \wedge \text{mam-strzałę}_t \wedge \neg\text{mam-strzałę}_{t+1}$$

Opisywanie akcji za pomocą formuł (2)

- Frame axioms (świat się za bardzo nie zmienia).
  Przykład:

$$\text{smok-śpi}_t \wedge \text{czytam-gazetę}_t \rightarrow \text{smok-śpi}_{t+1}$$

  (to dla wszystkich niewpływających na siebie par zmienna-stanu, akcja)
- Explanatory frame axioms

$$\text{smok-śpi}_t \wedge \neg\text{smok-śpi}_{t+1} \rightarrow (\text{rzucam-granat}_t \vee \text{gram-na-puzonie}_t \vee \ldots$$

- Informacja, że w każdym czasie wykonuję akcję (i tylko jedną – łatwa do zakodowania w CNF.

Algorytm SATPLAN

- Zasadniczo właśnie go opisaliśmy
- **Powtórka:** opisujemy świat, szukamy spełnialności formuły dla kolejnych T, jak znajdziemy, wypisujemy wartościowania, z którego odczytujemy sekwencję akcji.

Koniec części II

Sposób definiowania logiki (ogólnie)

Musimy podać 3 składniki:

1. **Składnię** – jak pisac formuły
2. **Semantykę** – co znaczą formuły, kiedy są prawdziwe
3. **Reguły wnioskowania** – jak z prawdziwych formuł wnioskować inne, również prawdziwe

Logiki mają różną siłę wyrazu i dają procedury o różnej złożoności (musimy **balansować** pomiędzy siłą wyrazu a obliczeniową trudnością logiki)

Reguły wnioskowania

Uwaga

Reguły wnioskowania dotyczą syntaktyki, nie semantyki.

Nasłynniejsza reguła wnioskowania

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

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

Można ją uogólnić do większej liczby przesłanek

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

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)

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 ($\mathcal{M}(KB) \subseteq \mathcal{M}(\phi)$).

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

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