Politechnika Poznańska
Instytut Informatyki

Adam Meissner
Adam.Meissner@put.poznan.pl
http://www.man.poznan.pl/~ameis

SZTUCZNA INTELIGENCJA

Problem spełnialności (SAT)

Literatura

[1] Cook S.A., *The complexity of theorem-proving procedures*, w: Proceedings of the third annual ACM symposium on Theory of computing, New York, 1971, s. 151–158.

[2] Davis M., Logemann G., Loveland D., *A machine program for theorem proving*, CACM, 1962, vol. 5, no. 7, s. 394–397.

[3] Howe J.M., King A., *A Pearl on SAT Solving in Prolog*, LNCS, 2010, vol. 6009, s. 165-174.

[4] Lifschitz V., *Reducing Graph Coloring to SAT*, http://www.cs.utexas.edu/users/vl/teaching/lbai/coloring.pdf (dostęp: 1.04.2020)

[5] Nadel A., *Understanding and Improving a Modern SAT Solver*, Tel Aviv University, 2009.

[6] The international SAT Competitions web page, http://www.satcompetition.org/ (dostęp: 1.04.2020).

Wprowadzenie (1)

Pojęcia podstawowe

- **zmienna zdaniowa (zmienna)** to bezargumentowy symbol predykatowy; interpretacja nadaje zmiennej zdaniowej wartość logiczną (prawda, 0 albo fałsz, 1)
- **literał pozytywny** to zmienna zdaniowa; **literał negatywny** to negacja zmiennej zdaniowej
- **klauzula** to alternatywa literałów; w szczególności klauzulą jest też pojedynczy literał oraz formuła pusta
- dowolna interpretacja spełniająca wszystkie klauzule z danego zbioru S jest nazywana **modelem** zbioru S, co oznacza się jako $m(S)$
- **problem SAT** polega na rozstrzygnięciu, czy dany zbiór klauzul S ma model lub na znalezieniu modelu $m(S)$; każdy model $m(S)$ nazywa się **rozwiązaniem** odpowiedniego problemu SAT.

Uwaga: do zdefiniowanego wyżej zbioru klauzul można przekształcić dowolny zbiór formuł logiki pierwszego rzędu, utworzonych ze zmiennych zdaniowych.

Przykład 1

Elementami zbioru S są następujące klauzule:

$\sim X_1 \vee X_2 \vee \sim X_3$

$\sim X_1 \vee \sim X_2 \vee X_3$

$X_1 \vee \sim X_2 \vee \sim X_3$

$X_1 \vee X_2 \vee X_3$

Przykładowy model zbioru S ma postać $m(S) = \{ X_1, X_2, X_3\}$.

Wprowadzenie (2)

Właściwości problemu SAT

- problem SAT jest pierwszym zidentyfikowanym problemem NP-zupełnym [1]
- podstawowym algorytmem rozwiązywania problemów SAT jest procedura DPLL [2]
- do rozwiązywania problemów SAT używa się programów zwanych **SAT-solverami** (ang. *SAT-solvers*)
- współczesne SAT-solvery mogą skutecznie rozwiązywać, na komputerach klasy PC, problemy obejmujące setki tysięcy klauzul i miliony zmiennych zdaniowych
- wiele problemów można przekształcić do postaci problemów SAT, następnie rozwiązać je, a uzyskany model odwzorować na rozwiązanie problemu źródłowego; proces konstruowania tych odwzorowań nosi nazwę odpowiednio **kodowania SAT** (ang. *SAT encoding*) albo **dekodowania SAT** (ang. *SAT decoding*)
- zbiór klauzul odpowiadający danemu problemowi jest deklaratywną reprezentacją tego problemu; oznacza to, że związek między danymi wejściowymi, a wyjściowymi występującymi w problemie jest relacyjny
- rozwiązywanie problemów przez kodowanie SAT stosuje się m.in. w takich dziedzinach jak:
  - szeregowanie zadań
  - planowanie działań
  - konstruowanie i weryfikowanie układów cyfrowych
  - kryptologia
  - bioinformatyka

Procedura DPLL (1)

Słownik

- **implikacja** L/b to przypisanie literałowi L wartości logicznej b; $\sim 0 = 1$ i $\sim 1 = 0$; jeżeli L jest literałem negatywnym to zmienna zdaniowa Z tworząca ten literał ma przypisaną wartość $\sim b$
- **literał nieokreślony** to literał, któremu nie przypisano żadnej wartości logicznej
- **literał** L jest **czysty** w zbiorze S, jeżeli występuje wśród klauzul ze zbioru S co najmniej jednokrotnie, a żadna klauzula nie zawiera literału $\sim L$
- **klauzula unarna** to klauzula zawierająca dokładnie jeden literał nieokreślony, a pozostałe literały (o ile istnieją) mają przypisaną wartość 0
- **decyzja** to implikacja L/b umieszczone na stosie M, decyzja jest L/b **końcowa** jeżeli wcześniej na stosie umieszczono implikację $L/\sim b$
- **konflikt** polega na skonstruowaniu implikacji L/b, podczas gdy wcześniej podjęto decyzję $L/\sim b$

Procedura DPLL (2)

Wejście: zbiór klauzul S, zbiór S_Z zawierający wszystkie zmienne zdaniowe występujące w zbiorze S, pusty stos M.

Wyjście: odpowiedź *true* jeżeli zbiór S ma model albo odpowiedź *false* w przeciwnym przypadku.

Krok 1: (wykrywanie literałów czystych). Utworzyć zbiór S' składający się z klauzul ze zbioru S zawierających co najmniej jeden wspólny literał czysty. Następnie, usunąć elementy zbioru S' ze zbioru S. Powtarzać wymienione czynności aż do przetworzenia wszystkich zbiorów S'.

Krok 2: (podejmowanie decyzji). Jeżeli zbiór S_Z jest pusty, to zakończyć działanie z odpowiedzią *true*, w przeciwnym przypadku wybrać (i usunąć) ze zbioru S_Z zmienną Z poczym utworzyć przypisanie Z/b. Następnie, umieścić to przypisanie na stosie decyzji M.

Krok 3: (propagowanie ograniczeń). Jeżeli zbiór S zawiera klauzulę unarną z nieokreślonym literałem L, to utworzyć implikację $L/1$ i usunąć zmienną zdaniową tworzącą literał L ze zbioru S_Z. Powtarzać czynności wymienione wcześniej w tym kroku aż do przetworzenia wszystkich klauzul unarnych albo do wystąpienia konfliktu. Powiązać wszystkie implikacje utworzone w tym kroku z decyzją znajdującą się na szczycie stosu M. W przypadku braku konfliktu przejść do kroku 2.

Krok 4: (rozwiązywanie konfliktów). Sprawdzić, czy stos M jest pusty. Jeżeli tak, to zakończyć działanie z odpowiedzią *false*. Jeżeli nie, to unieważnić wszystkie implikacje powiązane z decyzją Z/b, znajdującą się na szczycie stosu M. Następnie sprawdzić, czy decyzja Z/b jest końcowa. Jeżeli tak, to usunąć tę decyzję ze stosu i powtórzyć wszystkie czynności wymienione wcześniej w tym kroku. Jeżeli nie, to zastąpić decyzję Z/b przez decyzję $Z/\sim b$, oznaczyć tę decyzję jako końcową i przejść do kroku 3.

SAT-solvery (1)

Format DIMACS

- jest to popularny standard reprezentowania danych wejściowych dla SAT-solverów
- dane wejściowe znajdują się w pliku tekstowym o następującej strukturze:

```
c <Komentarz>
...
c <Komentarz>
p cnf <LiczbaZmiennych> <LiczbaKlauzul>
<ReprezentacjaKlauzuli>
...
<ReprezentacjaKlauzuli>
```

- wiersze zawierające komentarze są opcjonalne
- reprezentacja klauzuli ma postać następującą:
  - literał pozytywny reprezentuje dodatnia liczba całkowita
  - literał negatywny reprezentuje ujemna liczba całkowita
  - na końcu klauzuli znajduje się liczba 0
  - liczby w klauzuli są rozdzielone znakiem spacji

Przykład 2

Zbiór S z przykładu 1 ma następującą reprezentację w formacie DIMACS:

```
p cnf 3 4
-1 2 -3 0
-1 -2 3 0
1 -2 -3 0
1 2 3 0
```

SAT-solvery (2)

Przykładowe implementacje (*offline*)

- MiniSAT (zwycięzca zawodów SAT'2005 w kilku kategoriach)
  http://minisat.se/
- RSat (zwycięzca zawodów SAT'2007 w kategorii *industrial*)
  http://reasoning.cs.ucla.edu/rsat/home.html
- Lingeling (zwycięzca zawodów SAT'2014 w kilku kategoriach)
  http://fmv.jku.at/lingeling/

Przykładowe implementacje (*online*)

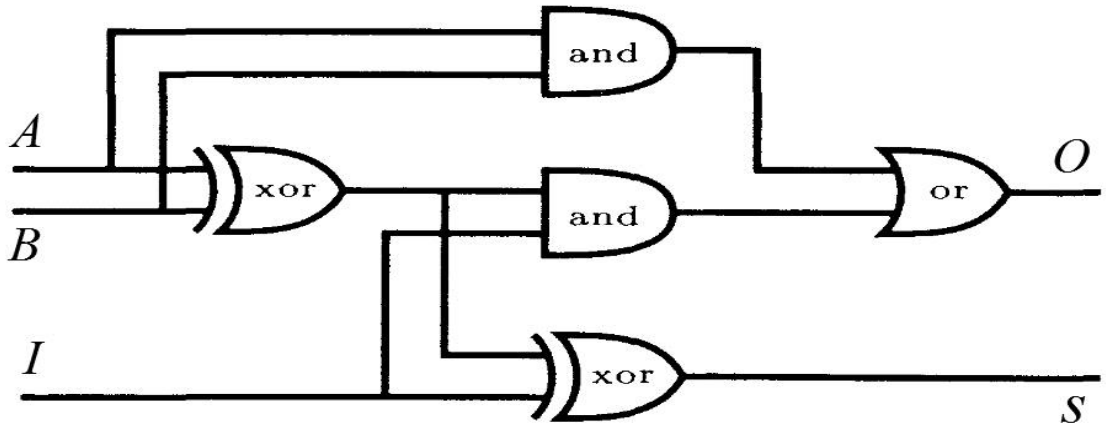
- Logictools (dobre do celów edukacyjnych)
  http://logictools.org/
- MiniSat in your browser
  https://msoos.github.io/cryptominisat_web/
- SATRennesPA
  http://satrennespa.irisa.fr/WebContent/

Przykładowe translatory CSP → SAT

- BEE Equi-Propagation Encoder
  http://amit.metodi.me/research/bee/
- Sugar: a SAT-based Constraint Solver
  http://bach.istc.kobe-u.ac.jp/sugar/

Przykładowe problemy SAT (1)

Sumator 1-bitowy z przeniesieniem



$$S \leftrightarrow A \oplus B \oplus I$$

$$O \leftrightarrow (A \wedge B) \vee (I \wedge (A \oplus B))$$

Wynik przekształcenia ww. formuł do postaci klauzulowej:

```
~S ∨ A ∨ B ∨ I
~S ∨ ~B ∨ ~A ∨ I
~S ∨ ~I ∨ ~A ∨ B
~S ∨ ~I ∨ ~B ∨ A
~A ∨ B ∨ I ∨ S
~B ∨ A ∨ I ∨ S
~I ∨ A ∨ B ∨ S
~I ∨ ~B ∨ ~A ∨ S
~O ∨ A ∨ I
~O ∨ A ∨ B
~O ∨ B ∨ I
~A ∨ ~B ∨ O
~I ∨ ~A ∨ B ∨ O
~I ∨ ~B ∨ A ∨ O
```

Przykładowe problemy SAT (2)

Sumator 1-bitowy z przeniesieniem (cd.)

Znaleźć wszystkie wartości wejściowe, dla których $S = 0$, a $O = 1$. Przy tym założeniu, zbiór wejściowy obejmuje następujące klauzule:

```
~A ∨ B ∨ I
~B ∨ A ∨ I
~I ∨ A ∨ B
~I ∨ ~B ∨ ~A
A ∨ I
A ∨ B
B ∨ I
```

Format DIMACS:

```
p cnf 3 7
-1 2 3 0
-2 1 3 0
-3 1 2 0
-3 -2 -1 0
1 3 0
1 2 0
2 3 0
```

Wejściowy zbiór klauzul ma następujące 3 modele: $\{A, B, \sim I\}$, $\{A, \sim B, I\}$, $\{\sim A, B, I\}$.

Przykładowe problemy SAT (3)

Kolorowanie mapy

Dana jest przykładowa mapa polityczna.

Rozpatruje się fragment mapy, który obejmuje następujące kraje oraz ich wszystkich sąsiadów:

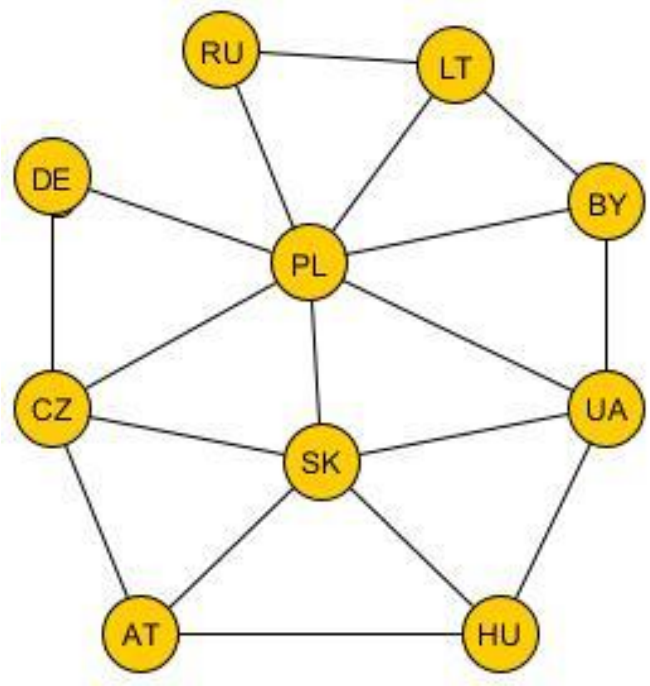
- Polska
- Czechy
- Słowacja

Pokolorować dany fragment mapy za pomocą co najwyżej czterech barw w taki sposób, aby żadne dwa kraje sąsiednie nie miały tej samej barwy.

Przykładowe problemy SAT (4)

Kolorowanie mapy (cd.)

Rozpatrywany fragment mapy można przedstawić w postaci następującego grafu.



Kodowanie SAT problemu 4-kolorowania grafu [4]

A *k-coloring* of a graph is a labelling of its vertices with at most k colors such that no two vertices sharing the same edge have the same color. The problem of generating a k-coloring of a graph (V, E) can be reduced to SAT as follows. For every $v \in V$ and every $i \in \{1, \ldots, k\}$, introduce an atom p_{vi}. Intuitively, this atom expresses that vertex v is assigned color i. Consider the following propositional formulas:

$$\begin{aligned} &\bigvee_{1 \le i \le k} p_{vi} && (v \in V), \\ &\neg(p_{vi} \wedge p_{vj}) && (v \in V,\ 1 \le i < j \le k), \\ &\neg(p_{vi} \wedge p_{wi}) && (\{v, w\} \in E,\ 1 \le i \le k). \end{aligned} \tag{1}$$

The interpretations satisfying these formulas are in a 1–1 correspondence with k-colorings of (V, E).

Przykładowe problemy SAT (5)

Kolorowanie mapy (cd.)

Poniższe klauzule stanowią fragment kodowania dotyczący Rosji. Zmienna *KR-i* oznacza *i*-ty kolor kraju *KR* dla *i*=1,4.

RU-1 ∨ RU-2 ∨ RU-3 ∨ RU-4
-RU-1 ∨ -RU-2
-RU-1 ∨ -RU-3
-RU-1 ∨ -RU-4
-RU-2 ∨ -RU-3
-RU-2 ∨ -RU-4
-RU-3 ∨ -RU-4

-LT-1 ∨ -RU-1
-LT-2 ∨ -RU-2
-LT-3 ∨ -RU-3
-LT-4 ∨ -RU-4
-PL-1 ∨ -RU-1
-PL-2 ∨ -RU-2
-PL-3 ∨ -RU-3
-PL-4 ∨ -RU-4

Rozwiązanie