Analiza Algoritmilor
Tema 2 – $\texttt{CSAT} \leq_P \texttt{CNFSAT}$

Termen de predare: **14.Jan.2024, 23:50**

Responsabili temă:
Alexandru-Petru TOADER,
Cezar-Stelian ZLATEA,
Mihai-Valentin DUMITRU

Data publicării: *11.Dec.2023*
Ultima actualizare: *11.Dec.2023*

Pentru cea de-a doua temă, va trebui să implementați programatic o reducere de la problema CSAT la CNFSAT.

1 CSAT

Un *circuit boolean* este un arbore în care:

- fiecare nod este o poartă logică (AND, OR sau NOT)
- copiii unui nod sunt inputurile porții respective
- părintele nodului este outputul porții
- rădăcina este o variabilă booleană – *outputul circuitului*
- frunzele sunt variabile booleene – *inputurile circuitului*

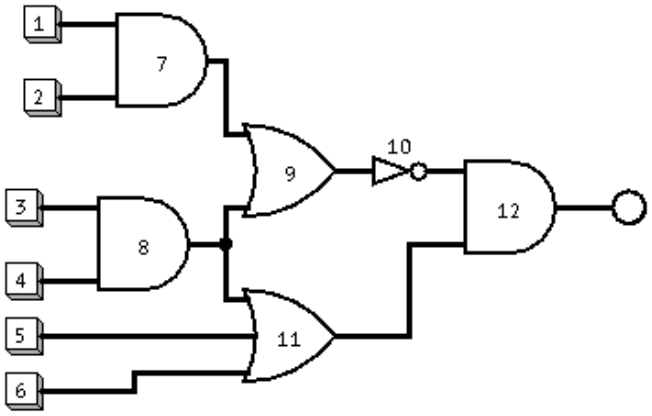

Figure 1: Exemplu de circuit boolean

Deși circuitele par doar o reprezentare diferită a unei formule booleene, există o diferență: circuitele pot să *refolosească* o valoare. Observați cum, în exemplu, poarta AND cu eticheta "8" e input pentru două porți OR.

Un circuit este *satisfiabil* dacă există o asociere între inputuri și valori booleene, astfel încât outputul să fie TRUE.

Problema de decizie CSAT primește ca input un circuit boolean; răspunsul este TRUE dacă circuitul este satisfiabil, FALSE altfel.

2 CNF SAT

O formulă logică ϕ, ce constă în variabile booleene și operații logice între acestea, este *satisfiabilă* dacă există o interpretare a variabilelor (o asociere de TRUE/FALSE pentru fiecare variabilă) astfel încât formula să fie adevărată.

În contextul temei, o să lucrăm exclusiv cu formule în *forma normală conjunctivă*.

O formulă în formă normală conjunctivă constă într-o *conjuncție* de una sau mai multe *clauze*. O clauză constă într-o disjuncție de unul sau mai mulți *literali*. Un literal este fie o variabilă, fie negarea unei variabile.

Exemplu de formulă CNF:

$$(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_4}) \wedge (x_2 \vee x_3 \vee \overline{x_4})$$

3 CSAT $\leq_P$ CNF SAT

O posibilă reducere polinomială de la CSAT la CNF SAT este următoarea formată din două părți:

1. transformă circuitul într-o formulă SAT

2. aplică reducerea SAT $\leq_P$ CNF SAT, studiată la curs

Pentru cea de-a doua reducere, găsiți o descriere completă în notițele de curs. Vom descrie aici prima reducere, CSAT $\leq_P$ SAT:

- crează o variabilă x_i pentru fiecare input i al circuitului
- pornind de la variabila de output a circuitului, aplică recursiv următorii pași:
  - **caz de bază**: dacă variabila curentă este un input i al circuitului, returnează subformula x_i
  - dacă variabila curentă este outputul unei porți NOT, returnează subformula $\overline{E}$, unde E e formula obținută prin aplicarea recursivă a acestei proceduri pe inputul porții
  - dacă variabila curentă este outputul unei porți AND cu m inputuri, returnează subformula $E_1 \wedge E_2 \wedge E_3 \wedge ... \wedge E_m$, unde E_i este formula obținută prin aplicarea recursivă a acestei proceduri peste al i-lea input al porții
  - dacă variabila curentă este outputul unei porți OR cu m inputuri, returnează subformula $E_1 \vee E_2 \vee E_3 \vee ... \vee E_m$, unde E_i este formula obținută prin aplicarea recursivă a acestei proceduri peste al i-lea input al porții

Aplicarea acestui algoritm peste circuitul din figura 1 rezultă în formula:

$$\overline{(x_1 \wedge x_2) \vee (x_3 \wedge x_4)} \wedge ((x_3 \wedge x_4) \vee x_5 \vee x_6)$$

4 Cerință

Va trebui să implementați programatic o reducere de la CSAT la SAT. Reducerea implementată poate fi cea descrisă în secțiunea precedentă, sau orice altă reducere corectă. Dacă implementați o altă reducere:

- dacă ați citit despre reducere dintr-o altă sursă, va trebui să o menționați în README.
- dacă este o reducere proprie, va trebui să schițați, în README, demonstrația corectitudinii

Programul vostru va fi invocat cu doi parametrii din linia de comandă:

1. numele unui fișier de intrare, care conține descrierea circuitului C
2. numele unui fișier de output, în care va trebui să scrieți formula rezultată din transformare

În continuare, prezentăm formatele de input și output.

5 Format input

Pentru a putea specifica circuitul, vom da nume nu doar inputurilor și outputului circuitului, ci și outputurilor tuturor porților; astfel, fiecare din acestea va fi identificat de o *variabilă*.

Într-un circuit cu n variabile, fiecare variabilă va fi identificată de un număr de la 1 la n, cu convenția că:

- outputul este mereu variabila n
- cele i inputuri sunt identificate cu numere de la 1 la i

Pe prima linie a inputului, se găsesc două numere:

- i, numărul de inputuri ale circuitului
- n, indexul outpului

Urmează apoi mai multe linii, fiecare descriind una din porțile circuitului. Fiecare linie constă în mai multe elemente separate prin câte un spațiu:

- descrierea porții: AND, OR sau NOT

- variabilele ce reprezintă inputurile porții
- variabila ce reprezintă outputul porții

Circuitele vizate în cadrul temei vor avea **maxim 100 de porți**; fiecare poartă va avea **maxim 60 de inputuri**.

Circuitul din figura 1 este reprezentat astfel:

```
6 12
AND 1 2 7
AND 3 4 8
OR 7 8 9
NOT 9 10
OR 8 5 6 11
AND 10 11 12
```

Atenție 1. Sunteți liberi să alegeți orice fel de reprezentare internă a circuitului.

6 Format output

Pentru a reprezenta o instanță de SAT, vom folosi formatul DIMACS, folosit în competiții de SAT-solving.

Prima linie va începe cu cuvintele cheie "p cnf", urmate de numărul de variabile n și de numărul de clauze m, separate de câte un spațiu. Variabilele poartă nume de la 1 la n.

Următoarele m linii reprezintă fiecare câte o clauză. Linia i conține până la n numere întregi corespunzând literalilor care compun clauza i, separate de câte un spațiu. Dacă un literal apare în forma negată, numărul respectiv va fi precedat de un minus "-".

Fiecare linie se termină cu un 0 (separat printr-un spațiu de ultimul literal).

Exemplu:

Formula: $(x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_3)$

Va fi reprezentată astfel:

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

7 Checker și teste automate

Pentru a facilita dezvoltarea temei, vă punem la dispoziție un checker automat și o suită de teste. Checkerul va invoca programul scris de voi pe fiecare test în parte. Rezultatul reducerii voastre este apoi trimis ca input către un SAT solver.

SAT solverul folosit la temă este `cadical-rel-1.5.3`, câștigătorul ediției din 2023 a "The International SAT Competition"[†] – o competiție anuală, afiliată cu una dintre conferințele de top din domeniu.[‡]

Solverul se găsește în directorul cu același nume; în subfolderul `bin/`, există un executabil precompilat, `cadical`, care va fi invocat de checker. În caz că executabilul nu funcționează pe mașina voastră, puteți să-l recompilați. Din directorul `cadical-rel-1.5.3`, rulați:

```
./configure
make
cp build/cadical bin/cadical
```

Există 30 de teste, fiecare test valorând 3 puncte. Pentru fiecare test există o limită de trei secunde de rulare; limita se referă strict la realizarea reducerii polinomiale, nu și la timpul petrecut de SAT solver pentru a determina satisfiabilitatea formulei generate.

[†] https://satcompetition.github.io/2023/results.html
[‡] http://satisfiability.org/SAT23/

În folderul `in/` se găsesc cele 30 de teste, cu un nume de forma `dataX.in`, unde `X` e numărul testului. Pentru fiecare test, există un fișier corespunzător în directorul `ref/`, cu un nume de forma `dataX.ref`, ce conține un singur "1", dacă graful conține o clică de dimensiunea căutată, respectiv un singur "0", dacă graful nu conține o clică de dimensiunea căutată.

Pentru a rula checkerul pe toate testele, este suficient să rulați:

```
./checker.sh
```

Puteți să rulați selectiv pe unul sau mai multe teste (e.g. pe testele 2, 7, 15), menționându-le ca argumente din linia de comandă:

```
./checker.sh 2 7 15
```

8 Punctaj

Tema valorează 10 puncte din totalul de 60 de puncte de parcurs. Tema va fi punctată de la 0 la 100, după următoarele criterii.

- 90 de puncte pentru testele automate.
- 10 puncte vor fi acordate în urma unei examinări manuale ce are ca scop evaluarea claritătii implementării și a prezentării acesteia în README. Tema nu este axată pe bune practici de dezvoltare software; deci deși vă recomandăm cu căldură să urmăriți un coding-style consecvent și restrictiv, nu se vor scădea puncte pentru aspecte particulare (e.g. linii prea lungi, funcții prea lungi etc.) Codul trebuie să fie însă inteligibil, în caz contrar se vor putea scădea puncte după de caz.

Atenție 2. Scopul testelor este să ajute cu dezvoltarea cerinței și cu punctarea temei.
Trebuie însă să rezolvați *cerința*, indiferent de teste. În cazul în care implementarea voastră nu respectă cerința, punctajul oferit de checker poate fi anulat total sau parțial în urma examinării manuale.

9 Trimitere temă

Va trebui să trimiteți următoarele:

- tot codul sursă necesar pentru implementarea temei
- un fișier `Makefile`
- un fișier `README`

Codul sursă poate consta în oricâte fișiere, grupate în orice fel de ierarhie de directoare. Puteți rezolva tema **în orice limbaj de programare doriți**.

Fișierul `Makefile` va trebui să conțină cel puțin următoarele două reguli:

- `build`: checkerul va rula prima dată `make build`
- `run`: checkerul va rula `make run INPUT=<inputfile> OUTPUT=<outputfile>`, după pasul de `build`.

Aveți în scheletul de cod un exemplu de `Makefile` care preia argumentele și le trimite unui program.

Dacă folosiți un limbaj necompilat (e.g. Python), puteți să trimiteți un `Makefile` în care regula de build nu face nimic.

În fișierul `README` ar trebui să descrieți, high-level, implementarea voastră. În caz că ați folosit o altă reducere decât cea menționată aici în enunț, va trebui să precizați asta în README.

Acestea trebuiesc puse într-o arhivă `.zip` cu numele:

IDLDAP_Grupa_Tema2.zip

Unde IDLDAP este ID-ul cu care vă logați pe moodle.

De exemplu:

mihai.dumitru2201_321CB_Tema2.zip

Fișierele `Makefile` și `README` trebuie să fie în rădăcina arhivei (nu într-un director în arhivă); `make build` va fi rulat după dezarhivare.

Tema va trebui încărcată pe moodle la această adresă.

9.1 Deadline

Tema trebuie trimisă până la data de **14 ianuarie 2023, ora 23:50**.

10 Forum

Pentru orice întrebări sau neclarități legate de conținutul temei, folosiți forumul temei. Evitați punerea de întrebări pe canale private (e.g. Teams, mail).

Nu postați pe forum fișierul cu implementarea voastră, sau fragmente din acesta.

11 Plagiat

Rezolvarea temei este individuală.

Copierea totală sau parțială a temei de la un alt student va rezulta în aplicare sancțiunilor menționate în regulament[§] pentru *toți studenții implicați*, fiind irelevant cine de la cine a luat.

Desigur, aveți voie să discutați între voi idei generale de soluții. Dar nu vă uitați pe *implementarea* unui coleg; nu preluați exemple concrete, nu cereți ajutor cu depanarea concretă a implementării voastre.

În situațiile în care nu sunteți siguri ce constituie un act de plagiat, întrebați pe forumul temei.

[§] https://ocw.cs.pub.ro/ppcarte/doku.php?id=aa:intro:rules