Formalizacija, implementacija i primene SAT rešavača

Filip Marić*

*Matematički fakultet
Beograd

Odbrana doktorske disertacije, 11. 6. 2009.

Pregled

Pregled

Pregled

SAT problem

Definicija (SAT problem)

Problem iskazne zadovoljivosti (SAT problem) je problem ispitivanja postojanja dodele vrednosti iskaznim promenljivim (valuacija) koja čini datu (KNF) iskaznu formulu tačnom. Zadovoljavajuće valuacije nazivamo modelima formule.

Primer

Formula

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

je tačna u modelu $\{x_1, \neg x_2, \neg x_3\}$.

Primer

Formula

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

nije zadovoljiva.

Pregled

Pristupi rešavanju SAT problema

Potpuni algoritmi - za svaku instancu problema mogu ispitati da li postoji model.

Stohastički algoritmi - ne mogu da pokažu da model ne postoji, ali mogu veoma brzo da nađu modele za neke velike, zadovoljive instance.

Razmatrani su isključivo potpuni algoritmi.

Pristupi rešavanju SAT problema

Potpuni algoritmi - za svaku instancu problema mogu ispitati da li postoji model.

Stohastički algoritmi - ne mogu da pokažu da model ne postoji, ali mogu veoma brzo da nađu modele za neke velike, zadovoljive instance.

Razmatrani su isključivo potpuni algoritmi.

Istorija automatskog rešavanja SAT problema

- Klasična DPLL (Davis-Putnam-Logemann-Loveland) procedura (1960., 1962.).
- Dokazana NP kompletnost (Cook, 1971.).
- Spektakularni napredak je načinjen od sredine 1990-tih.
- Moguće je rešiti neke formule koje sadrže $\approx 10\,000$ promenljivih i $\approx 1\,000\,000$ klauza (mada postoje i značajno manje formule koje nije moguće rešiti).

Razlozi za ovakav uspeh

- Konceptualna unapređenja klasične DPLL procedure (povratni skokovi, analiza konflikata, učenje, otpočinjanje iznova, . . .)
- Bolja implementacija (efikasne strukture podataka. . .)
- Heurističke komponente (strategije izbora literala, strategije zaboravljanja, . . .)

Primene SAT rešavača

Zahvaljujući ogromnom napretku u razvoju SAT rešavača, danas se rešavanje mnogih praktičnih problema se može svesti na rešavanje SAT problema. Problemi su najčešće iz oblasti:

- dizajna elektronskih kola,
- verifikacije softvera i hardvera,
- veštačke inteligencije,
- planiranja,
- operacionih istraživanja,
- konstrukcije kombinatornih objekata, ...

Opisi SAT rešavača

Konkretni opisi - Obično dati u obliku programskog (pseudo)koda. Veoma bliski realnim implementacijama, ali teški za razumevanje i rezonovanje o njima.

Apstraktni opisi - Obično dati u obliku sistema promena stanja. Laki za razumevanje, formalizaciju i rezonovanje o njima, ali kriju važne implementacione detalje.

Pregled

Klasična DPLL procedura - rekurzivna formulacija

```
function dpll (F : Formula) : (SAT, UNSAT)
begin
  if F is empty then                                          BAZA
    return SAT
  else if there is an empty clause in F then
    return UNSAT
  else there is a unit clause [l] in F then                   ZAKLJUČIVANJE
    return dpll(F[l → ⊤])
  else if there is a pure literal l in F then
    return dpll(F[l → ⊤])
  else begin                                                  PRETRAGA
    select a literal l occurring in F
    if dpll(F[l → ⊤]) = SAT then
      return SAT
    else
      return dpll(F[l → ⊥])
  end
end
```

Klasična DPLL procedura - rekurzivna formulacija

```
function dpll (F : Formula) : (SAT, UNSAT)
begin
  if F is empty then                                          BAZA
    return SAT
  else if there is an empty clause in F then
    return UNSAT
  else there is a unit clause [l] in F then                   ZAKLJUČIVANJE
    return dpll(F[l → ⊤])
  else if there is a pure literal l in F then
    return dpll(F[l → ⊤])
  else begin                                                  PRETRAGA
    select a literal l occurring in F
    if dpll(F[l → ⊤]) = SAT then
      return SAT
    else
      return dpll(F[l → ⊥])
  end
end
```

Klasična DPLL procedura - rekurzivna formulacija

```
function dpll (F : Formula) : (SAT, UNSAT)
begin
  if F is empty then                                          BAZA
    return SAT
  else if there is an empty clause in F then
    return UNSAT
  else there is a unit clause [l] in F then                   ZAKLJUČIVANJE
    return dpll(F[l → ⊤])
  else if there is a pure literal l in F then
    return dpll(F[l → ⊤])
  else begin                                                  PRETRAGA
    select a literal l occurring in F
    if dpll(F[l → ⊤]) = SAT then
      return SAT
    else
      return dpll(F[l → ⊥])
  end
end
```

Klasična DPLL procedura - rekurzivna formulacija

```
function dpll (F : Formula) : (SAT, UNSAT)
begin
  if F is empty then                                          BAZA
    return SAT
  else if there is an empty clause in F then
    return UNSAT
  else there is a unit clause [l] in F then                   ZAKLJUČIVANJE
    return dpll(F[l → ⊤])
  else if there is a pure literal l in F then
    return dpll(F[l → ⊤])
  else begin                                                  PRETRAGA
    select a literal l occurring in F
    if dpll(F[l → ⊤]) = SAT then
      return SAT
    else
      return dpll(F[l → ⊥])
  end
end
```

Modifikovana klasična DPLL procedura - rekurzivna formulacija

```
function dpll (M : Valuation) : (SAT, UNSAT)
begin
  if M ⊨¬ F then                                                    BAZA
    return UNSAT
  else if M is total wrt. variables of F
    return SAT
  else there is a unit clause in F wrt. M then          ZAKLJUČIVANJE
   (i.e. there a clause l ∨ l1 ∨ ... ∨ lk, st. l, l̄ ∉ M, l̄1, ... l̄k ∈ M)
    return dpll(M ∪ {l})
  else begin                                                        PRETRAGA
    select a literal l st. l ∈ F, l, l̄ ∉ M
    if dpll(M ∪ {l}) = SAT then
      return SAT
    else
      return dpll(M ∪ {l̄})
  end
end
```

Modifikovana klasična DPLL procedura - rekurzivna formulacija

```
function dpll (M : Valuation) : (SAT, UNSAT)
begin
  if M ⊨¬ F then                                              BAZA
    return UNSAT
  else if M is total wrt. variables of F
    return SAT
  else there is a unit clause in F wrt. M then          ZAKLJUČIVANJE
   (i.e. there a clause l ∨ l1 ∨ ... ∨ lk, st. l, l̄ ∉ M, l̄1, ... l̄k ∈ M)
    return dpll(M ∪ {l})
  else begin                                                  PRETRAGA
    select a literal l st. l ∈ F, l, l̄ ∉ M
    if dpll(M ∪ {l}) = SAT then
      return SAT
    else
      return dpll(M ∪ {l̄})
  end
end
```

Modifikovana klasična DPLL procedura - rekurzivna formulacija

```
function dpll (M : Valuation) : (SAT, UNSAT)
begin
  if M ⊨¬ F then                                                    BAZA
    return UNSAT
  else if M is total wrt. variables of F
    return SAT
  else there is a unit clause in F wrt. M then          ZAKLJUČIVANJE
   (i.e. there a clause l ∨ l1 ∨ ... ∨ lk, st. l, l̄ ∉ M, l̄1, ... l̄k ∈ M)
    return dpll(M ∪ {l})
  else begin                                                      PRETRAGA
    select a literal l st. l ∈ F, l, l̄ ∉ M
    if dpll(M ∪ {l}) = SAT then
      return SAT
    else
      return dpll(M ∪ {l̄})
  end
end
```

Modifikovana klasična DPLL procedura - rekurzivna formulacija

```
function dpll (M : Valuation) : (SAT, UNSAT)
begin
  if M ⊨¬ F then                                                    BAZA
    return UNSAT
  else if M is total wrt. variables of F
    return SAT
  else there is a unit clause in F wrt. M then          ZAKLJUČIVANJE
   (i.e. there a clause l ∨ l1 ∨ ... ∨ lk, st. l, l̄ ∉ M, l̄1, ... l̄k ∈ M)
    return dpll(M ∪ {l})
  else begin                                                        PRETRAGA
    select a literal l st. l ∈ F, l, l̄ ∉ M
    if dpll(M ∪ {l}) = SAT then
      return SAT
    else
      return dpll(M ∪ {l̄})
  end
end
```

Pregled

Označene valuacije

Uvode se u cilju nerekurzivne implementacije DPLL procedure.

Definicija (Označena valuacija)

Označena valuacija je lista literala u kojoj su neki od literala označeni kao *pretpostavljeni literali*.

Primer

Označena valuacija M može biti $[+1, |-2, +6, |+5, -3, +4, |-7]$.

Formalni sistem Krstića i Goela [KG07]

Decide:
$$\frac{l \in L \qquad l, \bar{l} \notin M}{M := M\ l}$$

UnitPropagate:
$$\frac{l \vee l_1 \vee \ldots \vee l_k \in F \qquad \bar{l}_1, \ldots, \bar{l}_k \in M \qquad l, \bar{l} \notin M}{M := M\ l}$$

Conflict:
$$\frac{C = no\_cflct \qquad \bar{l}_1 \vee \ldots \vee \bar{l}_k \in F \qquad l_1, \ldots, l_k \in M}{C := \{l_1, \ldots, l_k\}}$$

Explain:
$$\frac{l \in C \qquad l \vee \bar{l}_1 \vee \ldots \vee \bar{l}_k \in F \qquad l_1, \ldots, l_k \prec l}{C := C \cup \{l_1, \ldots, l_k\} \setminus \{l\}}$$

Learn:
$$\frac{C = \{l_1, \ldots, l_k\} \qquad \bar{l}_1 \vee \ldots \vee \bar{l}_k \notin F}{F := F \cup \{\bar{l}_1 \vee \ldots \vee \bar{l}_k\}}$$

Backjump:
$$\frac{C = \{l, l_1, \ldots, l_k\} \qquad \bar{l} \vee \bar{l}_1 \vee \ldots \vee \bar{l}_k \in F \qquad \text{level } l > m \geq \text{level } l_i}{C := no\_cflct \qquad M := M^{[m]}\ \bar{l}}$$

Forget:
$$\frac{C = no\_cflct \qquad c \in F \qquad F \setminus c \vDash c}{F := F \setminus c}$$

Restart:
$$\frac{C = no\_cflct}{M := M^{[0]}}$$

Stanje rešavača - (F, M, C)

- F - formula
- M - označena valuacija
- C - klauza analize konflikata

Ilustracija - uprošćeni sistem

DPLL pretraga

Stanje rešavača

- M - označena valuacija

Decide:

$$\frac{l \in F \quad l, \bar{l} \notin M}{M := M \mid l}$$

UnitPropagate:

$$\frac{l \vee l_1 \vee \ldots \vee l_k \in F \qquad \bar{l}_1, \ldots, \bar{l}_k \in M \qquad l, \bar{l} \notin M}{M := M\ l}$$

Backtrack:

$$\frac{M \models \neg F \quad M = M' \mid l\, M'' \quad \text{decisions } M'' = []}{M := M'\ \bar{l}}$$

Ilustracija - uprošćeni sistem

DPLL pretraga

Stanje rešavača

- M - označena valuacija

Decide:

$$\frac{l \in F \quad l, \bar{l} \notin M}{M := M \mid l}$$

UnitPropagate:

$$\frac{l \vee l_1 \vee \ldots \vee l_k \in F \qquad \bar{l}_1, \ldots, \bar{l}_k \in M \qquad l, \bar{l} \notin M}{M := M\ l}$$

Backtrack:

$$\frac{M \models \neg F \quad M = M' \mid l\, M'' \quad \text{decisions } M'' = []}{M := M'\ \bar{l}}$$

Ilustracija - uprošćeni sistem

DPLL pretraga

Stanje rešavača

- M - označena valuacija

Decide:

$$\frac{l \in F \quad l, \bar{l} \notin M}{M := M \mid l}$$

UnitPropagate:

$$\frac{l \vee l_1 \vee \ldots \vee l_k \in F \qquad \bar{l}_1, \ldots, \bar{l}_k \in M \qquad l, \bar{l} \notin M}{M := M\ l}$$

Backtrack:

$$\frac{M \models \neg F \quad M = M' \mid l\ M'' \quad \text{decisions } M'' = []}{M := M'\ \bar{l}}$$

Ilustracija - uprošćeni sistem

DPLL pretraga

Stanje rešavača

- M - označena valuacija

Decide:

$$\frac{l \in F \quad l, \bar{l} \notin M}{M := M \mid l}$$

UnitPropagate:

$$\frac{l \vee l_1 \vee \ldots \vee l_k \in F \qquad \bar{l}_1, \ldots, \bar{l}_k \in M \qquad l, \bar{l} \notin M}{M := M\ l}$$

Backtrack:

$$\frac{M \models \neg F \quad M = M' \mid l\ M'' \quad \text{decisions } M'' = [\,]}{M := M'\ \bar{l}}$$

Ilustracija - uprošćeni sistem

DPLL pretraga

Stanje rešavača

- M - označena valuacija

Decide:

$$\frac{l \in F \quad l, \bar{l} \notin M}{M := M \,|\, l}$$

UnitPropagate:

$$\frac{l \vee l_1 \vee \ldots \vee l_k \in F \qquad \bar{l}_1, \ldots, \bar{l}_k \in M \qquad l, \bar{l} \notin M}{M := M \; l}$$

Backtrack:

$$\frac{M \models \neg F \quad M = M' \,|\, l \, M'' \quad \text{decisions } M'' = []}{M := M' \; \bar{l}}$$

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	sat?	M
	UNDEF	$[]$
Decide (l = $+1$)	UNDEF	$[\|+1]$
UnitProp (c = $[-1, +2]$, l = $+2$)	UNDEF	$[\|+1, +2]$
UnitProp (c = $[-1, -3]$, l = -3)	UNDEF	$[\|+1, +2, -3]$
Decide (l = $+4$)	UNDEF	$[\|+1, +2, -3, \|+4]$
UnitProp (c = $[-4, +5]$, l = $+5$)	UNDEF	$[\|+1, +2, -3, \|+4, +5]$
Backtrack ($M \models \neg [+3, -4, -5]$)	UNDEF	$[\|+1, +2, -3, -4]$
UnitProp (c = $[-2, +4, +5]$, l = $+5$)	UNDEF	$[\|+1, +2, -3, -4, +5]$
$M \not\models \neg F$, (vars M) = (vars F)	SAT	$[\|+1, +2, -3, -4, +5]$

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	sat?	M
	UNDEF	[]
Decide (l = +1)	UNDEF	[|+1]
UnitProp (c = [−1, +2], l = +2)	UNDEF	[|+1, +2]
UnitProp (c = [−1, −3], l = −3)	UNDEF	[|+1, +2, −3]
Decide (l = +4)	UNDEF	[|+1, +2, −3, |+4]
UnitProp (c = [−4, +5], l = +5)	UNDEF	[|+1, +2, −3, |+4, +5]
Backtrack ($M \models \neg$ [+3, −4, −5])	UNDEF	[|+1, +2, −3, −4]
UnitProp (c = [−2, +4, +5], l = +5)	UNDEF	[|+1, +2, −3, −4, +5]
$M \not\models \neg F$, (vars M) = (vars F)	SAT	[|+1, +2, −3, −4, +5]

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	sat?	M
	UNDEF	$[]$
Decide (l = +1)	UNDEF	$[\|+1]$
UnitProp (c = $[-1, +2]$, l = +2)	UNDEF	$[\|+1, +2]$
UnitProp (c = $[-1, -3]$, l = −3)	UNDEF	$[\|+1, +2, -3]$
Decide (l = +4)	UNDEF	$[\|+1, +2, -3, \|+4]$
UnitProp (c = $[-4, +5]$, l = +5)	UNDEF	$[\|+1, +2, -3, \|+4, +5]$
Backtrack ($M \models \neg [+3, -4, -5]$)	UNDEF	$[\|+1, +2, -3, -4]$
UnitProp (c = $[-2, +4, +5]$, l = +5)	UNDEF	$[\|+1, +2, -3, -4, +5]$
$M \not\models \neg F$, (vars M) = (vars F)	SAT	$[\|+1, +2, -3, -4, +5]$

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	sat?	M
	UNDEF	[]
Decide (l = +1)	UNDEF	[|+1]
UnitProp (c = [−1, +2], l = +2)	UNDEF	[|+1, +2]
UnitProp (c = [−1, −3], l = −3)	UNDEF	[|+1, +2, −3]
Decide (l = +4)	UNDEF	[|+1, +2, −3, |+4]
UnitProp (c = [−4, +5], l = +5)	UNDEF	[|+1, +2, −3, |+4, +5]
Backtrack ($M \models \neg$ [+3, −4, −5])	UNDEF	[|+1, +2, −3, −4]
UnitProp (c = [−2, +4, +5], l = +5)	UNDEF	[|+1, +2, −3, −4, +5]
$M \not\models \neg F$, (vars M) = (vars F)	SAT	[|+1, +2, −3, −4, +5]

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	sat?	M
	UNDEF	[]
Decide (l = +1)	UNDEF	[|+1]
UnitProp (c = [−1, +2], l = +2)	UNDEF	[|+1, +2]
UnitProp (c = [−1, −3], l = −3)	UNDEF	[|+1, +2, −3]
Decide (l = +4)	UNDEF	[|+1, +2, −3, |+4]
UnitProp (c = [−4, +5], l = +5)	UNDEF	[|+1, +2, −3, |+4, +5]
Backtrack ($M \models \neg$ [+3, −4, −5])	UNDEF	[|+1, +2, −3, −4]
UnitProp (c = [−2, +4, +5], l = +5)	UNDEF	[|+1, +2, −3, −4, +5]
$M \not\models \neg F$, (vars M) = (vars F)	SAT	[|+1, +2, −3, −4, +5]

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	sat?	M
	UNDEF	[]
Decide (l = +1)	UNDEF	[|+1]
UnitProp (c = [−1, +2], l = +2)	UNDEF	[|+1, +2]
UnitProp (c = [−1, −3], l = −3)	UNDEF	[|+1, +2, −3]
Decide (l = +4)	UNDEF	[|+1, +2, −3, |+4]
UnitProp (c = [−4, +5], l = +5)	UNDEF	[|+1, +2, −3, |+4, +5]
Backtrack ($M \models \neg$ [+3, −4, −5])	UNDEF	[|+1, +2, −3, −4]
UnitProp (c = [−2, +4, +5], l = +5)	UNDEF	[|+1, +2, −3, −4, +5]
$M \not\models \neg F$, (vars M) = (vars F)	SAT	[|+1, +2, −3, −4, +5]

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	sat?	M
	UNDEF	[]
Decide (l = +1)	UNDEF	[|+1]
UnitProp (c = [−1, +2], l = +2)	UNDEF	[|+1, +2]
UnitProp (c = [−1, −3], l = −3)	UNDEF	[|+1, +2, −3]
Decide (l = +4)	UNDEF	[|+1, +2, −3, |+4]
UnitProp (c = [−4, +5], l = +5)	UNDEF	[|+1, +2, −3, |+4, +5]
Backtrack ($M \models \neg [+3, -4, -5]$)	UNDEF	[|+1, +2, −3, −4]
UnitProp (c = [−2, +4, +5], l = +5)	UNDEF	[|+1, +2, −3, −4, +5]
$M \not\models \neg F$, (vars M) = (vars F)	SAT	[|+1, +2, −3, −4, +5]

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	sat?	M
	UNDEF	[]
Decide (l = +1)	UNDEF	[|+1]
UnitProp (c = [−1, +2], l = +2)	UNDEF	[|+1, +2]
UnitProp (c = [−1, −3], l = −3)	UNDEF	[|+1, +2, −3]
Decide (l = +4)	UNDEF	[|+1, +2, −3, |+4]
UnitProp (c = [−4, +5], l = +5)	UNDEF	[|+1, +2, −3, |+4, +5]
Backtrack ($M \models \neg$ [+3, −4, −5])	UNDEF	[|+1, +2, −3, −4]
UnitProp (c = [−2, +4, +5], l = +5)	UNDEF	[|+1, +2, −3, −4, +5]
$M \not\models \neg F$, (vars M) = (vars F)	SAT	[|+1, +2, −3, −4, +5]

Primer

$F = [[-1, +2], [-1, -3], [-2, +4, +5], [+3, -4, -5], [-4, +5]]$

Primenjeno pravilo	sat?	M
	UNDEF	$[]$
Decide (l = $+1$)	UNDEF	$[\|+1]$
UnitProp (c = $[-1, +2]$, l = $+2$)	UNDEF	$[\|+1, +2]$
UnitProp (c = $[-1, -3]$, l = -3)	UNDEF	$[\|+1, +2, -3]$
Decide (l = $+4$)	UNDEF	$[\|+1, +2, -3, \|+4]$
UnitProp (c = $[-4, +5]$, l = $+5$)	UNDEF	$[\|+1, +2, -3, \|+4, +5]$
Backtrack ($M \models \neg [+3, -4, -5]$)	UNDEF	$[\|+1, +2, -3, -4]$
UnitProp (c = $[-2, +4, +5]$, l = $+5$)	UNDEF	$[\|+1, +2, -3, -4, +5]$
$M \not\models \neg F$, (vars M) = (vars F)	SAT	$[\|+1, +2, -3, -4, +5]$

Ključna pitanja za implementaciju

- Prilikom izmene valuacije M, kako efikasno otkriti klauze iz F koje su postale netačne i/ili jedinične? Neophodno je korišćenje specijalizovanih kompleksnih struktura podataka (npr. shema dva posmatrana literala).
- Kako odabrati literal za pravilo Decide? Neophodno je koristiti pametne heuristike koje ovo kontrolišu.

Ključna pitanja za implementaciju

- Prilikom izmene valuacije M, kako efikasno otkriti klauze iz F koje su postale netačne i/ili jedinične? Neophodno je korišćenje specijalizovanih kompleksnih struktura podataka (npr. shema dva posmatrana literala).
- Kako odabrati literal za pravilo Decide? Neophodno je koristiti pametne heuristike koje ovo kontrolišu.

Ključna pitanja za implementaciju

- Prilikom izmene valuacije M, kako efikasno otkriti klauze iz F koje su postale netačne i/ili jedinične? Neophodno je korišćenje specijalizovanih kompleksnih struktura podataka (npr. shema dva posmatrana literala).
- Kako odabrati literal za pravilo Decide? Neophodno je koristiti pametne heuristike koje ovo kontrolišu.

Pregled

Postojeće implementacije SAT rešavača

- Postoji veliki broj efikasnih implementacija.
- Postoji veliki broj implementacija otvorenog koda.
- Na SAT takmičenjima, vrši se eksperimentalna evaluacija rešavača.
- Najuspešniji i najuticajniji rešavači: SATO, Chaff, Berkmin, MiniSat, Picosat, Rsat, ...

Pristup implementaciji

- Iako većina savremenih SAT rešavača radi po principima koji su opisani apstraktnim opisima, njihov kôd veoma retko ovo jasno oslikava.
- Algoritmi i heuristike su obično „tvrdo-kodirani" i izmešani u okviru koda.
- Često jedna funkcija ili modul koda implementira više različitih algoritama i heuristika.
- Često je implementacija nekog algoritma ili heuristike rasprostrta kroz više funkcija u kodu.

Softverski dizajn rešavača se može unaprediti.

Pristup implementaciji

- Iako većina savremenih SAT rešavača radi po principima koji su opisani apstraktnim opisima, njihov kôd veoma retko ovo jasno oslikava.
- Algoritmi i heuristike su obično „tvrdo-kodirani” i izmešani u okviru koda.
- Često jedna funkcija ili modul koda implementira više različitih algoritama i heuristika.
- Često je implementacija nekog algoritma ili heuristike rasprostrta kroz više funkcija u kodu.

Softverski dizajn rešavača se može unaprediti.

Cilj implementacije

Napraviti implementaciju koja eksplicitno prati apstraktni formalni opis, ali sadrži i sve implementacione tehnike niskog nivoa. U okviru koda jasno razdvojiti sve različite koncepte SAT rešavanja.

Pregled

Opis implementacije jezgra rešavača u imperativnom pseudo-jeziku.

- Pseudo-jezik podseća na jezik pascal.
- Opisano je DPLL jezgro savremenih SAT rešavača.
- Kod eksplicitno prati apstraktne opise preko sistema prelazaka stanja.
- Heurističke komponente su identifikovane i jasno izdvojene.
- Dato je nekoliko različitih opisa, sa različitim nivoima detaljnosti.
- Kôd je formalno verifikovan korišćenjem Horove logike.

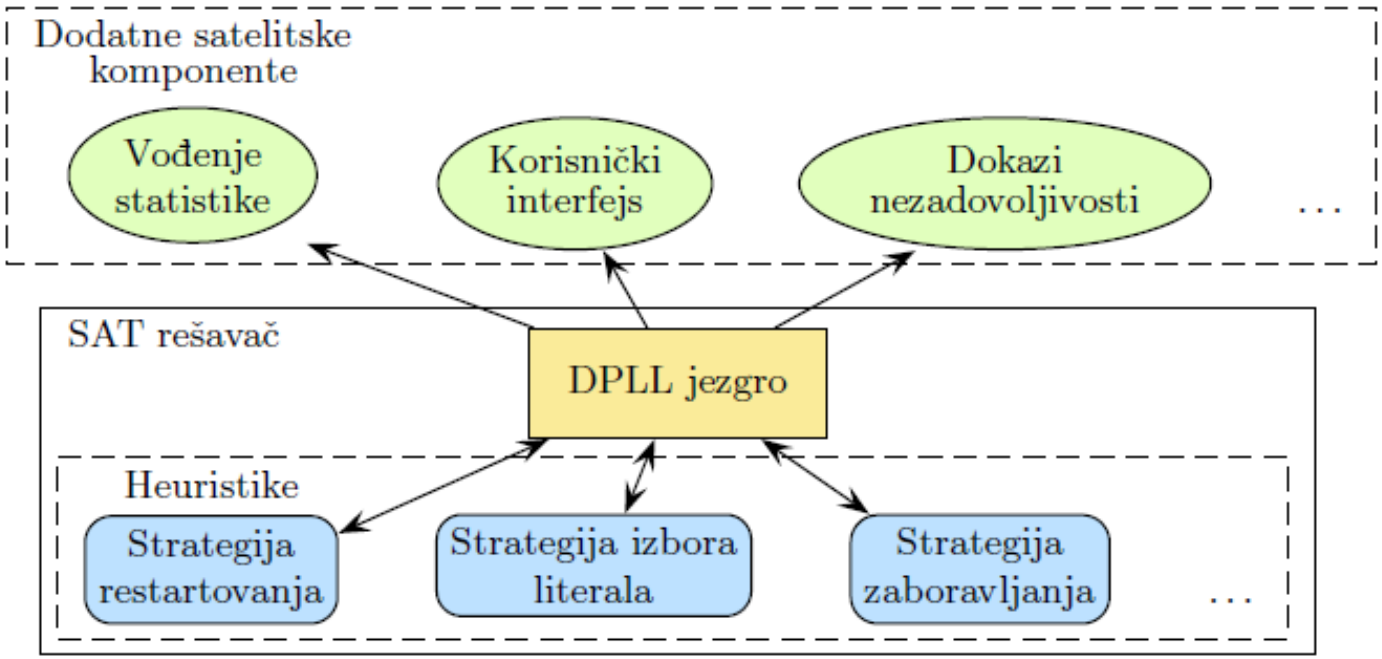
Primer koda

```
function applyDecide()
begin
    l := selectLiteral();
    assertLiteral(l, true)
end
```

Pregled

- Na osnovu verifikovanog jezgra napravljen je rešavač ArgoSAT.
- Softver otvorenog koda (`http://argo.matf.bg.ac.rs`).
- Objektno-orijentisana fleksibilna arhitektura.
- Značajno oslanjanje na projektne obrasce.
- Implementacija na jeziku C++.

Jezgro rešavača i satelitske komponente



Eksperimentalni rezultati

- Iako efikasnost nije bila primarni cilj, dobijeni rezultati su zadovoljavajući.
- Rešavač ArgoSAT je evaluiran na korpusima sa SAT takmičenja iz 2002., 2007. i SAT utrke 2009.
- Pošto ArgoSAT podržava različite heuristike, teško je oceniti koji rezultati su relevantni za poređenje sa ostalim sistemima.

Poređenje sa rešavačem MiniSat

- ArgoSAT 1.0 sa podrazumevanim parametrima.
- MiniSat 2.0 - preprocesiranje formula i napredne tehnike minimalizacije konfliktih klauza isključene.
- Korpus SATRace 2008. - 100 instanci.
- Dopušteno 20min po instanci.

	ArgoSAT	MiniSat
Broj rešenih formula	79	80
Središnje vreme	93.2s	69.9s

Poređenje sa rešavačem MiniSat

- ArgoSAT 1.0 sa podrazumevanim parametrima.
- MiniSat 2.0 - preprocesiranje formula i napredne tehnike minimalizacije konfliktih klauza isključene.
- Korpus SATRace 2008. - 100 instanci.
- Dopušteno 20min po instanci.

	ArgoSAT	MiniSat
Broj rešenih formula	79	80
Središnje vreme	93.2s	69.9s

Pregled

Motivacija

- Rešavači zasnovani na DPLL proceduri daju samo SAT/UNSAT odgovore, i pitanje je u kojoj meri je moguće pouzdati se u njih.
- Realne primene zahtevaju izuzetno visok stepen pouzdanosti.

Cilj

Obezbediti SAT rešavače u čije se rezultate možemo pouzdati.

Motivacija

- Rešavači zasnovani na DPLL proceduri daju samo SAT/UNSAT odgovore, i pitanje je u kojoj meri je moguće pouzdati se u njih.
- Realne primene zahtevaju izuzetno visok stepen pouzdanosti.

Cilj

Obezbediti SAT rešavače u čije se rezultate možemo pouzdati.

Pristupi dostizanju pouzdanosti

Pristupi

1. Proširivanje rešavače mogućnošću generisanja dokaza njihovih tvrdnji i provera dokaza nezavisnim proveravačima u čiju se pouzdanost možemo pouzdati.
2. Primeni formalne metode i verifikovati same SAT rešavače.

U ovoj tezi je primenjen drugi pristup.

Pristupi dostizanju pouzdanosti

Pristupi

1. Proširivanje rešavače mogućnošću generisanja dokaza njihovih tvrdnji i provera dokaza nezavisnim proveravačima u čiju se pouzdanost možemo pouzdati.
2. Primeni formalne metode i verifikovati same SAT rešavače.

U ovoj tezi je primenjen drugi pristup.

Prednosti formalne verifikacije

- Izbegava se dodatno vreme i prostor koji je potreban za generisanje, skladištenje i proveru objektnih dokaza.
- Doprinosi se boljem teorijskom razumevanju kako i zašto SAT rešavači funkcionišu.
- Verifikovani SAT rešavači mogu da posluže za proveru rezultata drugih neverifikovanih sistema (BDD, provera modela, SMT rešavači, ...).
- Neki delovi verifikovanih SAT rešavača mogu da doprinesu efikasnijoj proveri objektnih dokaza za SAT.
- Demonstrira se kako je zahvaljujući napretku u tehnologiji verifikacije softvera došlo vreme kada je moguće formalno verifikovati veoma kompleksne softverske sisteme.

Slojevi formalne verifikacije SAT rešavača

Verifikovano je nekoliko različitih opisa SAT rešavača uz korišćenje odgovarajućih verifikacijskih paradigmi (svake sa svojim prednostima i manama).

1. Izvršena je potpuna formalizacija nekoliko različitih sistema prelaska stanja za SAT i dokazana je njihova korektnost.
2. Izvršena je verifikacija implementacije SAT rešavača urađene na imperativnom pseudojeziku korišćenjem Horove logike.
3. Izvršena je implementacija SAT rešavača u okviru logike višeg reda, dokazana je njena korektnost i izvršena je ekstrakcija koda u nekoliko funkcionalnih jezika.

Mehanički verifikovani dokazi

- Kako bi se postigao zadovoljavajući, najviši, stepen pouzdanosti, dokaze je potrebno sprovesti u okviru nekog sistema za formalno dokazivanje teorema (eng. proof assistant).
- Celokupna formalizacija i verifikacija SAT rešavača je urađena u okviru sistema Isabelle/HOL.

Pregled

Sintaksa

Primer

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

Model: $\{x_1, \neg x_2, \neg x_3\}$

Isabelle tipovi

```
types    Variable  = nat
datatype Literal   = Pos Variable | Neg Variable
types    Clause    = "Literal list"
types    Formula   = "Clause list"

types    Valuation = "Literal list"
```

Semantika

Definicija

$v \models l$ ako i samo ako $l \in v$

```
literalTrue :: "Literal => Valuation => bool"
```

$v \models\neg l$ ako i samo ako $\overline{l} \in v$

```
literalFalse :: "Literal => Valuation => bool"
```

$v \models c$ ako i samo ako $\exists l.\ l \in c \wedge v \models l$

```
clauseTrue :: "Clause => Valuation => bool"
```

$v \models\neg c$ ako i samo ako $\forall l.\ l \in c \rightarrow v \models\neg l$

```
clauseFalse :: "Clause => Valuation => bool"
```

$v \models F$ ako i samo ako $\forall c.\ c \in F \rightarrow v \models c$

```
formulaTrue :: "Formula => Valuation => bool"
```

$v \models\neg F$ ako i samo ako $\exists c.\ c \in F \wedge v \models\neg c$

```
formulaFalse :: "Formula => Valuation => bool"
```

Semantka (nastavak)

Definicija

(consistent v) ako i samo ako ($\neg\exists l.\ v \vDash l \wedge v \vDash \bar{l}$)

```
consistent ::  "Valuation => bool"
```

(model v F) ako i samo ako (consistent $v \wedge v \vDash F$)

```
model ::  "Valuation => Formula => bool"
```

(sat F) ako i samo ako ($\exists v.$ model v F)

```
satisfiable ::  "Formula => bool"
```

Pregled

Relacija prelaska - formalna definicija

Definicija (Stanje)

Stanje (M, F) je uređeni par koji sadrži označenu valuaciju M i formulu F.

Definicija

$$\text{decide } (M_1, F_1)\ (M_2, F_2) \iff$$

$$\exists l. \quad l \in F_1 \ \wedge\ l \notin M_1 \ \wedge\ \bar{l} \notin M_1 \ \wedge\ M_2 = M_1 \mathbin{@} l^{\top} \ \wedge\ F_2 = F_1$$

Decide:

$$\frac{l \in F \quad l, \bar{l} \notin M}{M := M \mid l}$$

Relacija prelaska - formalna definicija

Definicija (Stanje)

Stanje (M, F) je uređeni par koji sadrži označenu valuaciju M i formulu F.

Definicija

$$\text{decide } (M_1, F_1)\ (M_2, F_2) \iff$$

$$\exists l. \quad l \in F_1 \ \wedge\ l \notin M_1 \ \wedge\ \bar{l} \notin M_1 \ \wedge\ M_2 = M_1 \mathbin{@} l^{\top} \ \wedge\ F_2 = F_1$$

Decide:

$$\frac{l \in F \quad l, \bar{l} \notin M}{M := M \mid l}$$

Relacija prelaska - formalna definicija

Definicija

$$(M_1, F_1) \rightarrow (M_2, F_2) \iff \begin{array}{l} \text{decide } (M_1, F_1)\ (M_2, F_2) \ \vee \\ \text{backtrack } (M_1, F_1)\ (M_2, F_2) \ \vee \\ \text{unitPropagate } (M_1, F_1)\ (M_2, F_2) \end{array}$$

Relacija $\rightarrow^*$ je *refleksivno tranzitivno zatvorenje* relacije $\rightarrow$.
Stanje $([], F_0)$ je početno za formulu F_0.
Stanje (M, F) je završno ako je minimalni element relacije $\rightarrow$, tj., ako ne postoji stanje (M', F') takvo da $(M, F) \rightarrow (M', F')$.

Definicija (Prihvatajuće stanje)

Stanje (M, F) je prihvatajuće ako:

1. nema konflikta (tj., $M \not\models \neg F$),
2. pravilo Decide ne može da se primeni (i.e., var $l \in$ vars F_0, $l \notin M$ and $\bar{l} \notin M$)

Definicija (Odbacujuće stanje)

Stanje (M, F) je odbacujuće ako:

1. postoji konflikt (tj., $M \models \neg F$),
2. pravilo Backtrack ne može da se primeni (tj., $(\text{decisions } M) = []$).

Definicija (Prihvatajuće stanje)

Stanje (M, F) je prihvatajuće ako:

1. nema konflikta (tj., $M \not\models \neg F$),
2. pravilo Decide ne može da se primeni (i.e., var $l \in$ vars F_0, $l \notin M$ and $\bar{l} \notin M$)

Definicija (Odbacujuće stanje)

Stanje (M, F) je odbacujuće ako:

1. postoji konflikt (tj., $M \models \neg F$),
2. pravilo Backtrack ne može da se primeni (tj., (decisions M) $= []$).

Teorema (Saglasnost)

Neka $([], F_0) \rightarrow^* (M, F)$.

- *Ako je stanje* (M, F) *prihvatajuće, tada je formula* F_0 *zadovoljiva i* M *je njen model (tj.,* sat F_0 *i* model M F_0*).*
- *Ako je stanje* (M, F) *odbacujuće, tada je formula* F_0 *nezadovoljiva (tj.,* $\neg$sat F_0*).*

Teorema (Saglasnost)

Neka $([], F_0) \rightarrow^* (M, F)$.

- *Ako je stanje* (M, F) *prihvatajuće, tada je formula* F_0 *zadovoljiva i* M *je njen model (tj.,* sat F_0 *i* model M F_0*).*
- *Ako je stanje* (M, F) *odbacujuće, tada je formula* F_0 *nezadovoljiva (tj.,* $\neg$sat F_0*).*

Teorema (Zaustavljanje)

Relacija $\longrightarrow$ je dobro-zasnovano uređenje, tj., ne postoji beskonačan opadajući lanac

$$([\,], F_0) \longrightarrow (M_1, F_1) \longrightarrow (M_2, F_2) \longrightarrow \ldots$$

Teorema (Potpunost)

Svako završno stanje (M, F) je ili prihvatajuće ili odbacujuće.

Kako se dokazuju ove teoreme?

Invarijante

$Invariant_{consistent}$: consistent M

$Invariant_{distinct}$: distinct M

$Invariant_{varsM}$: vars $M \subseteq$ vars F

$Invariant_{impliedLiterals}$: $\forall l.\ l \in M \implies (F \ @ \ \text{decisionsTo}\ l\ M) \models l$

Teorema

Ako $([], F_0) \rightarrow^* (M, F)$, *tada sve invarijante važe u stanju* (M, F).

Kako se dokazuju ove teoreme?

Zaustavljanje se dokazuje korišćenjem dobro-zasnovanih uređenja.

Definicija

$$l_1 \prec^{lit} l_2 \iff (\text{isDecision } l_1) \wedge \neg(\text{isDecision } l_2)$$

Definicija

$$M_1 \succ_{trail} M_2 \iff M_1 \prec^{lit}_{lex} M_2,$$

pri čemu je $\prec^{lit}_{lex}$ leksikografsko proširenje relacije $\prec^{lit}$.

Definicija

$$\begin{aligned} M_1 \succ^{r}_{trail} M_2 \iff\ & (\text{distinct } M_1) \wedge (\text{vars } M_1) \subseteq Vbl \\ & (\text{distinct } M_2) \wedge (\text{vars } M_2) \subseteq Vbl \wedge \\ & M_1 \succ_{trail} M_2 \end{aligned}$$

Pregled

Plitko utapanje u logiku višeg reda - osobine

Vrši se implementacija u okviru HOL koja se posmatra kao čist funkcionalni programski jezik.

Prednosti:

- Program se izražava kao skup rekurzivnih funkcija.
- Nema potrebe za korišćenjem specijalizovanih programskih logika (npr. Horove logike).
- Metodi koji se koriste prilikom dokaza su uglavnom indukcija i jednakosno rezonovanje.
- Izvršan kod je moguće automatski generisati.

Mane:

- Ne postoji mogućnost modifikovanja podataka „u mestu”.
- Bočni efekti nisu mogući.
- Globalne promenljive čine stanje rešavača koje je potrebno prenositi kao argument funkcijama.

Utapanje SAT rešavača

- Utapanje rešavača zasnovanog na klasičnoj DPLL proceduri je bilo jednostavna vežba [MJ09a].
- Utapanje savremenog SAT rešavača je predstavljalo veliki izazov [Mar09b].

Utapanje savremenog SAT rešavača

- Implementacija prati apstraktne prelaske stanja i imperativni pseudokod.
- Implementirane su sva značajna savremena unapređenja DPLL algoritma (povratni skokovi, analiza konflikata, učenje, zaboravljanje, otpočinjanje iznova).
- Implementirana je i većina implementacionih tehnika nižeg nivoa (shema dva posmatrana literala).

Stanje rešavača

```
record State =
   "getM"                 :: LiteralTrail
   "getF"                 :: Formula
   "getSATFlag"           :: ExtendedBool
   "getConflictFlag"      :: bool
   "getConflictClause"    :: pClause
   "getQ"                 :: "Literal list"
   "getReason"            :: "Literal ⇒ pClause option"
   "getWatch1"            :: "pClause ⇒ Literal option"
   "getWatch2"            :: "pClause ⇒ Literal option"
   "getWatchList"         :: "Literal ⇒ pClause list"
   "getC"                 :: Clause
   "getCl"                :: Literal
   "getCll"               :: Literal
```

Stanje rešavača

```
record State =
   "getM"                :: LiteralTrail
   "getF"                :: Formula
   "getSATFlag"          :: ExtendedBool
   "getConflictFlag"     :: bool
   "getConflictClause"   :: pClause
   "getQ"                :: "Literal list"
   "getReason"           :: "Literal ⇒ pClause option"
   "getWatch1"           :: "pClause ⇒ Literal option"
   "getWatch2"           :: "pClause ⇒ Literal option"
   "getWatchList"        :: "Literal ⇒ pClause list"
   "getC"                :: Clause
   "getCl"               :: Literal
   "getCll"              :: Literal
```

applyDecide – implementira pravilo Decide

```
definition applyDecide ::  "Variable set ⇒ State ⇒ State"
where
"applyDecide decisionVars =
   do
      l ← selectLiteral decisionVars;
      assertLiteral l True
   done
"
```

Glavna funkcija rešavača

```
definition solve_formula ::  "Formula ⇒ ExtendedBool StateTransformer"
where
"solve_formula F0 =
  do initialize F0;
     solve_loop (vars F0);
     readSATFlag
  done"
```

solve_loop – totalna rekurzija

```
function (domintros, tailrec)
  solve_loop ::  "Variable set ⇒ unit StateTransformer"
where
"solve_loop Vbl =
  do SATFlag ← readSATFlag;
    (if (SATFlag = UNDEF) then
      do solve_loop_body Vbl;
         solve_loop Vbl
      done)
  done"
by pat_completeness auto
```

Glavna petlja rešavača

```
definition solve_loop_body ::  "Variable set ⇒ unit StateTransformer"
where
"solve_loop_body decisionVars =
  do exhaustiveUnitPropagate;
     conflictFlag ← readConflictFlag; M ← readM;
     (if conflictFlag then
        (if (currentLevel M) = 0 then
           updateSATFlag FALSE
         else
           do applyConflict;
              applyExplainUIP;
              applyLearn;
              applyBackjump
           done)
      else
        (if (vars (elements M) ⊇ decisionVars) then
           updateSATFlag TRUE
         else
           applyDecide decisionVars))
  done
"
```

Totalna korektnost

Teorema

$$\text{solve } F_0 = SAT \iff \text{sat } F_0$$

Kako se dokazuje?

- Korišćeni su dokazi korektnosti za sisteme prelaska stanja.
- Uvedene su nove invarijante (24 zajedno sa starim).

Primer složene invarijante

$$\begin{aligned}\forall c.\ c < |F| \implies M \models \neg\,(watch_1\ c) \implies \\ (\exists l.\ l \in c \wedge M \models l \wedge \text{level}\ l \leq \text{level}\ \overline{(watch_1\ c)}) \vee \\ (\forall l.\ l \in c \wedge l \neq (watch_1\ c) \wedge l \neq (watch_2\ c) \implies \\ M \models \neg\, l \wedge \text{level}\ \bar{l} \leq \text{level}\ \overline{(watch_1\ c)}).\end{aligned}$$

- Parcijalno zaustavljaje funkcija definisanih generalnom rekurzijom je dokazano korišćenjem istih uređenja kao i za sisteme prelaska stanja.

Poređenje sa relevantnim radovima

- Apstraktni opisi su objavljeni sa neformalnim dokazima korektnosti.
- Formalna verifikacija urađena za klasičnu DPLL proceduru u sistemu Coq (Lescuyer, Conchon, 2008.).
- Formalna verifikacija urađena za savremeni SAT rešavač u sistemu PVS, ali bez mnogih važnih implementacionih detalja (Shankar, Vaucher, 2008.).
- U ovoj tezi su po prvi put verifikovani neki detalji rešavača od kojih je, svakako, najznačajnija shema dva posmatrana literala.

Neke brojke

- Oko 1 čovek godine rada.
- $\approx$ 25 000 linija Isabelle/Isar koda.
- Dokumenti u PDF formatu oko $\approx$ 700 strana.

Pregled

Razmatrane primene SAT rešavača

Autor je razmatrao tri polja primene SAT rešavača.

1. SMT rešavači
2. Parsiranje prirodnog jezika
3. Generisanje rasporeda časova

U nastavku će biti ukratko opisana primena SAT rešavača na automatsko raspoređivanje časova.

Razmatrane primene SAT rešavača

Autor je razmatrao tri polja primene SAT rešavača.

1. SMT rešavači
2. Parsiranje prirodnog jezika
3. Generisanje rasporeda časova

U nastavku će biti ukratko opisana primena SAT rešavača na automatsko raspoređivanje časova.

Pregled

Pristup

- Uslovi korektnosti i udobnosti rasporeda časova se kodiraju iskaznom formulom.
- SAT rešavač se koristi za pronalaženje zadovoljavajućih valuacija.
- Zadovoljavajuće valuacije predstavljaju ispravne rasporede časova.

Nedostatak

- Nedostatak: problem je formulisan kao problem *odlučivanja*, a ne kao problem *optimizacije*.
- Ograničenja se formulišu tako striktno, da svako rešenje koje se pronađe predstavlja raspored koji je dovoljno dobar kandidat za početak nastave.

Uslovi korektnosti

- Svaki čas iz unapred poznate liste časova mora biti raspoređen i to tačno jednom u rasporedu.
- Predavač ne može da predaje dva različita predmeta u isto vreme. Moguće je da se zahteva da predavač predaje isti predmet za nekoliko grupa studenata istovremeno.
- Grupa ne može da prisustvuje istovremeno na više različitih časova.
- U jednom sali u jednom terminu ne može da bude više od jednog predavača.

Uslovi udobnosti

Sistem omogućava definisanje veoma širokog spektra različitih uslova.

- Željeni i zabranjeni termini.
- Preklapanje grupa i nastavnika.
- Broj nastavnih dana.
- Trajanje nastavnog dana.
- Pauze u toku nastave.
- Uzastopni nastavni dani.
- ...

Uspešna primena

- Ova tehnika je do sada uspešno primenjena za izradu 12 rasporeda časova u jednoj srednjoj školi i na tri fakulteta u Beogradu.

Rezultati

Potrebno računarsko vreme:

- fakulteti - manje od 5 minuta
- srednja škola - nekoliko sati

Kvalitet rasporeda obično nadmašuje rasporede koji su ručno izrađivani.

Pregled

Doprinosi teze I

1. Opisani su detalji savremenih SAT rešavača na metodičan način koji čitaocima bez predznanja u ovoj oblasti omogućava njihovo razumevanje.
2. Opisan je apstraktni formalni okvir za izgradnju efikasnih i korektnih SAT rešavača.
3. Opisana je implementacija jezgra SAT rešavača u pseudokodu, tako da kôd prati opis algoritama višeg nivoa i jasno razdvaja koncepte prisutne u okviru rešavača (što ga čini jednostavnijim za razumevanje, održavanje, modifikaciju i dokazivanje korektnosti), a sadrži sve implementacione tehnike i trikove niskog nivoa (što ga čini efikasnim).

Doprinosi teze II

4. Formalno je verifikovana korektnost SAT rešavača opisanih kroz apstraktni formalni okvir i kroz konkretne implementacije.
5. Izgrađena je fleksibilna implementacija realnog savremenog SAT rešavača zasnovanog na opisanom jezgru i objektno-orijentisanoj paradigmi.
6. Demonstrirana je snaga savremenih SAT rešavača, kroz prikaz nekih od postojećih i razvoj nekih novih primena u rešavanju praktičnih problema.

Zaključak

Glavna poruka

Moguće izgraditi SAT rešavač tako da njegova implementacija eksplicitno prati apstraktni formalni opis, ali sadrži i sve implementacione tehnike niskog nivoa. Ovako izgrađen rešavač je moguće formalno verifikovati, a sa druge strane on može biti dovoljno efikasan i imati mnogobrojne praktične primene.


S. Krstic and A. Goel.
Architecting solvers for SAT modulo theories: Nelson-oppen with DPLL.
In *FroCos*, pp. 1–27, Liverpool, 2007.


R. Nieuwenhuis, A. Oliveras, and C. Tinelli.
Solving SAT and SAT Modulo Theories: From an Abstract Davis-Putnam-Logemann-Loveland Procedure to DPLL(T).
Journal of the ACM 53(6), 2006.


F. Marić.
Formalization and implementation of SAT solvers.
Journal of Automated Reasoning, 43(1), 2009.


F. Marić.
A formally verified SAT solver.
submitted to *TCS*.


F. Marić, Predrag Janičić.
Formal Correctness Proof for DPLL Procedure.
Informatica, 2009.


F. Marić, P. Janičić.
SAT Verification Project.
In *TPHOLS'09* - Emerging trends - TU Munchen technical report, 2009.


M. Nikolić, F. Marić, P. Janičić.
Instance-Based Selection of Policies for SAT Solvers.
In *SAT'09*, LNCS 5584, 2009.


M. Vujošević-Janičić, F. Marić, D. Tošić.
Using Simplex Method in Verifying Software Safety.
YUJOR, accepted for publication.