

Automatsko rezonovanje – beleške sa predavanja Primene SAT i SMT rešavača

Milan Banković (po slajdovima Filipa Marića)

* Matematički fakultet,
Univerzitet u Beogradu

Prolećni semestar 2026/27.

Pregled

1 Primene iskazne logike

Primene iskazne logike

Izražajnost iskazne logike

Jezik iskazne logike je veoma jednostavan:

- sve promenljive mogu uzimati samo vrednosti **tačno** ili **netačno**
- zbog toga se svaki problem mora modelovati u terminima logičkih vrednosti
- iskazi koji nama imaju neko intuitivno značenje se u iskaznoj logici razmatraju apstraktno
 - zbog toga se sve logičke veze među iskazima moraju izraziti logičkim veznicima
 - npr. važi $x < y \wedge y < z \Rightarrow x < z$, ali ako ove iskaze apstrahujemo iskaznim slovima p, q i r , moramo „ručno” da zadamo uslov $p \wedge q \Rightarrow r$, jer se to više ne podrazumeva
- ovo ponekad zahteva modelovanje na veoma niskom nivou, a dobijeni modeli mogu biti veoma glomazni
- ipak, veoma široka klasa problema se može predstaviti na ovaj način

Svođenje na SAT

Kada se problem svede na jezik iskazne logike, tada se rešavanje problema može svoditi na:

- ispitivanje tautologičnosti dobijene formule
- ispitivanje zadovoljivosti dobijene formule

U praksi se obično razmatra zadovoljivost, dok se problem tautologičnosti obično svodi na (ne)zadovoljivost negacije:

- razlog za ovo leži u izrazitoj efikasnosti modernih SAT rešavača
- glomaznost iskaznih modela obično nije problem za savremene SAT rešavače
- otuda je svođenje problema na SAT postalo izuzetno popularno u zadnje dve decenije

Dve vrste primena

SAT rešavači kao dokazivači valjanosti

Tipična primena je u verifikaciji formalnih sistema (najčešće hardvera):

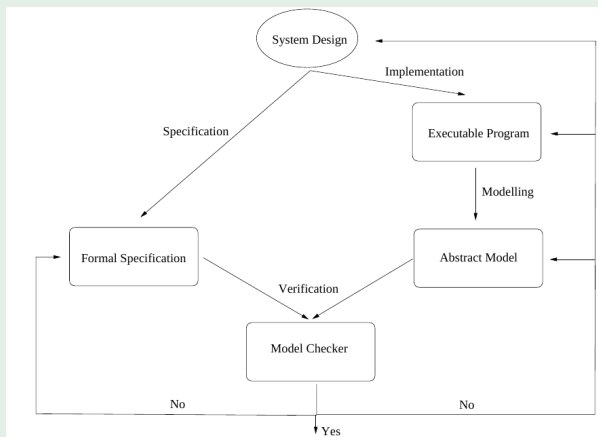
- rad sistema se predstavi iskaznom formulom P
- svojstvo koje mora da važi se predstavi formulom Q
- sada $P \Rightarrow Q$ mora da bude tautologija
- ekvivalentno, $P \wedge \neg Q$ mora da bude nezadovoljljiva

SAT rešavači i rešavanje problema kombinatorne pretrage

- uslovi koje rešenje problema mora da zadovoljava predstave se klauzama KNF formule
- zadovoljavajuća valuacija koju SAT rešavač pronalazi kodira traženo rešenje

Proveravanje modela

Strukturni prikaz



Proveravanje modela

Opis komponenti

- **Modelovanje**: proces formulisanja apstraktnog modela sistema na osnovu konkretne implementacije
- **Apstraktni model**: tranzicioni sistem (skup stanja i relacija prelaska nad tim skupom)
- **Formalna specifikacija**: precizan opis svojstava koje apstraktni model treba da zadovolji (na jeziku matematičke logike)
- **Proveravač modela**: alat koji automatski proverava da li apstraktni model zadovoljava formalnu specifikaciju
- U slučaju da ne zadovoljava, proveravač daje **kontraprimer** u vidu putanje u modelu koja narušava zadato svojstvo
- Kontraprimer se koristi za otkrivanje greške, korekciju modela i/ili specifikacije

Tranzicioni sistem

Tranzicioni sistem

Tranzicioni sistem je uređena petorka $(S, \longrightarrow, I, \mathcal{V}, \lambda)$, gde je:

- S skup stanja
- $\longrightarrow \subseteq S \times S$ relacija prelaska (u oznaci $s_i \longrightarrow s_j$)
- $I \subseteq S$ skup početnih stanja
- $\mathcal{V}$ skup iskaznih promenljivih (skup predikata)
- $\lambda : S \rightarrow \mathbb{P}\mathcal{V}$ funkcija mapiranja (svakom stanju pridružuje skup predikata koji važe u tom stanju)

Napomene

- Skup stanja može biti konačan ili beskonačan (tipično je konačan, ali veoma veliki)
- Bez ograničenja opštosti, pretpostavljamo da je relacija $\longrightarrow$ *totalna*, tj. $(\forall s \in S)(\exists s' \in S)(s \longrightarrow s')$
- Sistem može biti deterministički (ako $(\forall s \in S)(\exists! s' \in S)(s \longrightarrow s')$) i nedeterministički

Tranzicioni sistem

Putanja u sistemu

Putanja u sistemu $T = (S, \longrightarrow, I, \mathcal{V}, \lambda)$ je beskonačni niz stanja $\sigma = s_0 s_1 s_2 \dots$ takav da važi $s_i \longrightarrow s_{i+1}$ za svako $i \geq 0$. Pri tom ćemo koristiti oznaku $\sigma_i := s_i$ za i -to stanje putanje i $\sigma|_i := s_i s_{i+1} s_{i+2} \dots$ za „rep” putanje počev od i -tog stanja.

Izvršavanje u sistemu T je bilo koja putanja σ takva da je $\sigma_0 \in I$. Za stanje $s \in S$ kažemo da je dostižno ako postoji izvršavanje σ takvo da $s \in \sigma$.

Hardver kao tranzicioni sistem

Modelovanje hardvera kao tranzicionog sistema

- **Skup stanja** je određen stanjem registara (memorijskih komponenti, flip-floпова) u datom hardverskom sistemu i vrednostima bitova na ulazu
 - ulazi dolaze iz spoljašnjeg okruženja (druge komponente, interakcija sa korisnikom), i njihove vrednosti se ne mogu predvideti (**nedeterminizam**)
 - tipično imamo veliki broj stanja (eksponencijalan u odnosu na broj bitova registara i bitova ulaza)
 - određene početne konfiguracije hardvera se smatraju **početnim stanjima**
- **Relacija prelaska** opisuje ponašanje kola u svakom ciklusu časovnika
- **Skup predikata** $\mathcal{V}$ obično odgovara bitovima ulaza, bitovima registara, kao i bitovima izlaza
 - $\lambda(s)$ je tada skup svih bitova koji imaju vrednost 1 u stanju s

The diagram illustrates a Turing machine (TM) and its state transition graph. On the left, the TM is shown with an input tape containing the symbol x and a blank symbol $\square$. The machine has a head that can move left ($\leftarrow$) or right ($\rightarrow$). The machine's internal state is represented by a box labeled r . The machine's operations are defined by a set of rules: $\{x, \square\} \rightarrow \{y, \square\}$ and $\{x, r\} \rightarrow \{x, r, y\}$. The machine's initial state is $\{x=0, r=0\}$ and its final state is $\{x=1, r=1\}$. On the right, the state transition graph is shown, which is a directed graph with four states: $\{x=0, r=0\}$, $\{x=1, r=0\}$, $\{x=0, r=1\}$, and $\{x=1, r=1\}$. The transitions are: $\{x=0, r=0\} \rightarrow \{x=1, r=0\}$, $\{x=1, r=0\} \rightarrow \{x=1, r=1\}$, $\{x=0, r=1\} \rightarrow \{x=1, r=1\}$, and $\{x=1, r=1\} \rightarrow \{x=0, r=1\}$. The graph also shows self-loops on each state.

- x – jednobitni ulaz
- y – jednobitni izlaz
- r – jednobitni registar
- Stanje: par bitova (x, r)
- Početna stanja: za $r = 0$
- Graf relacije prelaska je prikazan na desnoj slici
- Ulaz x uvodi nedeterminizam
- $\mathcal{V} = \{x, r, y\}$
- Skup $\lambda(s)$ je prikazan uz svako stanje u grafu

Formalna specifikacija

Svojstva sistema

Formalna specifikacija predstavlja skup svojstava izraženih na nekom formalnom jeziku koja sistem mora da zadovoljava:

- svojstva se izražavaju u terminima predikata koji važe ili ne važe u dostižnim stanjima sistema
- svojstva se mogu odnositi na pojedinačna stanja: npr. „u svakom stanju važi svojstvo P ”
- svojstva se mogu odnositi i na više stanja odjednom: npr. „ako u nekom stanju važi P , onda u sledećem stanju mora važiti Q ”
- neka najčešće razmatrane vrste svojstava:
 - **invarijante** – svojstva koja moraju da važe u svim dostižnim stanjima sistema
 - **sigurnosna svojstva** – da se neka negativna pojava nikada neće desiti
 - **svojstva živosti** – neka pozitivna pojava će se sigurno desiti u nekom narednom stanju
 - **svojstva pravednosti** – neka pojava će se dešavati beskonačno puta u nastavku izvršavanja
- za izražavanje se tipično koristi neki vid **temporalne logike** koji omogućava zadavanje vremenskih odrednica:
 - npr. *važi u svakom dostižnom stanju, desiće se pre ili kasnije, važi počev od nekog stanja* i sl.
- pod određenim uslovima, svojstva se mogu aproksimirati iskaznim formulama:
 - npr. ne možemo reći „u svakom dostižnom stanju važi svojstvo P ”, ali možemo reći „u prvih k stanja izvršavanja važi svojstvo P ”

Opis na jeziku iskazne logike

Opis izvršavanja tranzicionog sistema $(S, \longrightarrow, I, \mathcal{V}, \lambda)$

Neka je $\mathcal{V} = \{p_1, \dots, p_n\}$ i neka je dato proizvoljno izvršavanje $s_0 s_1 s_2 \dots$ u tranzicionom sistemu.

- Za svako stanje s_i imaćemo skup iskaznih slova $p_1^i, \dots, p_n^i$ (kopije predikata u i -tom stanju)
- Skup početnih stanja opisujemo iskaznom formulom $I(s_0)$ nad promenljivama $p_1^0, \dots, p_n^0$:
 - npr. možemo imati formulu $\bigvee_{s \in I} \bigwedge_j I_j^s$, gde je $I_j^s = p_j^0$ ili $I_j^s = \neg p_j^0$, u zavisnosti od toga da li je $p_j \in \lambda(s)$
 - često postoji i jednostavniji način da se izraze početni uslovi
- Prelazak $s_i \longrightarrow s_{i+1}$ predstavljamo iskaznom formulom $R(s_i, s_{i+1})$ nad promenljivama $p_1^i, \dots, p_n^i, p_1^{i+1}, \dots, p_n^{i+1}$
 - npr. možemo imati formulu $\bigwedge_{s \in S} ((\bigwedge_j I_j^s) \Rightarrow (\bigvee_{s' \mid s \longrightarrow s'} (\bigwedge_j I_j^{s'})))$, gde je $I_j^s = p_j^i$ ili $I_j^s = \neg p_j^i$, a $I_j^{s'} = p_j^{i+1}$ ili $I_j^{s'} = \neg p_j^{i+1}$, u zavisnosti od toga da li predikat p_j važi u stanjima s , odnosno s' , respektivno
 - opet, u konkretnom sistemu, opis relacije prelaska može biti i jednostavniji

Ograničena provera modela

Ograničena provera modela (bounded model checking (BMC))

Ideja je da proverimo da li neko svojstvo važi u prvih k koraka izvršavanja:

- ispostavi se da se sva ranije spomenuta svojstva tada mogu izraziti na jeziku iskazne logike
- ovde ćemo razmatrati invarijante, jer su one najjednostavnije, budući da se odnose na pojedinačna stanja

Provera invarijante

Neka je invarijanta data iskaznom formulom $J(s)$ nad predikatima $p_1, \dots, p_n$ u stanju s . Da bismo proverili da li invarijanta mora da važi u prvih k koraka proizvoljnog izvršavanja, ispitujemo zadovoljivost sledeće formule:

$$I(s_0) \wedge R(s_0, s_1) \wedge \dots \wedge R(s_{k-1}, s_k) \wedge \neg(J(s_0) \wedge \dots \wedge J(s_k))$$

- ako je formula **nezadovoljiva**, tada invarijanta sigurno važi u prvih k koraka
- ako je formula **zadovoljiva**, tada zadovoljavajuća valuacija daje **kontraprimer**, tj. kodira izvršavanje u kome invarijanta neće važiti (ovo olakšava pronalaženje grešaka u modelu)

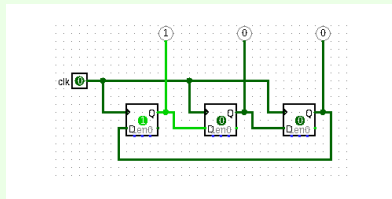
Napomena

Iako ograničena provera modela nije potpuna metoda, ona u praksi može otkriti veliki broj grešaka.

Primer

Primer

Na slici je dat je trobitni ciklični registar sa početnom vrednošću 100. Dokazati svojstvo: „U svakom trenutku je tačno jedan bit registra uključen“.



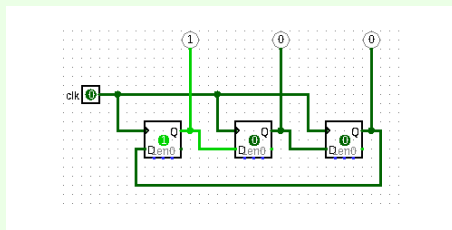
Imamo model:

- Skup stanja: $S = \{0, 1\}^3$ (sve trojke bitova pqr)
- Relacija prelaska: $pqr \longrightarrow rpq$
- Početno stanje: 100
- Predikati: $\mathcal{V} = \{p, q, r\}$
- $\lambda(s)$ sadrži jedini bit koji je uključen u stanju s

Primer (2)

Primer

(nastavak) Pretpostavimo da vršimo ograničenu proveru modela za $k = 3$.



- $I(s_0) \equiv (p^0 \wedge \neg q^0 \wedge \neg r^0)$ (početno stanje je 100)
- $R(s_i, s_{i+1}) \equiv (p^{i+1} \Leftrightarrow r^i) \wedge (q^{i+1} \Leftrightarrow p^i) \wedge (r^{i+1} \Leftrightarrow q^i)$
- Rad kola u prva 3 koraka se opisuje formulom:

$$I(s_0) \wedge R(s_0, s_1) \wedge R(s_1, s_2) \wedge R(s_2, s_3)$$

Primer

- $J(s_i) \equiv (p^i \vee q^i \vee r^i) \wedge (\neg p^i \vee \neg q^i) \wedge (\neg p^i \vee \neg r^i) \wedge (\neg q^i \vee \neg r^i)$
- Formula čija se zadovoljivost ispituje:
 $I(s_0) \wedge R(s_0, s_1) \wedge R(s_1, s_2) \wedge R(s_2, s_3) \wedge \neg(J(s_0) \wedge J(s_1) \wedge J(s_2) \wedge J(s_3))$

Dokazivanje indukcijom

Dokazivanje indukcijom

Jedan način da se obezbedi potpunost (tj. da invarijanta važi za svako dostižno stanje) je da se koristi **indukcija**. Baza indukcije glasi:

$$I(s_0) \wedge \neg J(s_0)$$

Intuitivno, nezadovoljivost ove formule znači da za svako početno stanje invarijanta J važi. Sada imamo induktivni korak:

$$J(s_0) \wedge R(s_0, s_1) \wedge \neg J(s_1)$$

Intuitivno, nezadovoljivost ove formule znači da kad god invarijanta J važi u nekom stanju s_0 , tada važi i u stanju s_1 koje može slediti za stanjem s_0 u datom tranzicionom sistemu.

Dokazivanje indukcijom (2)

k -indukcija

Ponekad je potrebno znati da je invarijanta važila u prethodnih k stanja da bismo mogli da zaključimo da važi i u sledećem stanju. Tada prethodna formula za bazu indukcije dobija oblik:

$$I(s_0) \wedge R(s_0, s_1) \wedge \dots \wedge R(s_{k-2}, s_{k-1}) \wedge \neg(J(s_0) \wedge \dots \wedge J(s_{k-1}))$$

Intuitivno, nezadovoljivost ove formule znači da će u prvih k stanja proizvoljnog izvršavanja invarijanta J sigurno važiti (ovo je ista formula koju smo imali i za BMC). Induktivni korak dobija sledeći oblik:

$$J(s_0) \wedge J(s_1) \wedge \dots \wedge J(s_{k-1}) \wedge R(s_0, s_1) \wedge \dots \wedge R(s_{k-1}, s_k) \wedge \neg J(s_k)$$

Intuitivno, nezadovoljivost ove formule znači da ako invarijanta važi u prethodnih k stanja, tada važi i u sledećem stanju.

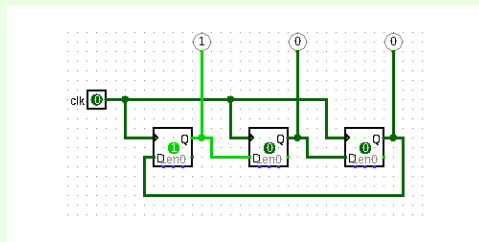
Napomena

- u slučaju da je formula za bazu indukcije zadovoljiva, tada imamo kontraprimer dužine k
- u slučaju da je formula za induktivni korak zadovoljiva, tada je potrebno probati za veće k

Primer

Primer

Primenimo indukciju na prethodni primer:



U ovom primeru, indukcija će proći za $k = 1$, tj. imaćemo nezadovoljivost formule $I(s_0) \wedge \neg J(s_0)$, kao i formule $J(s_0) \wedge R(s_0, s_1) \wedge \neg J(s_1)$.

Problem zadovoljenja ograničenja

Definicija

Problem zadovoljenja ograničenja (engl. *Constraint Satisfaction Problem (CSP)*) je uređena trojka $\mathcal{P} = (\mathcal{X}, \mathcal{D}, \mathcal{C})$, pri čemu je:

- $\mathcal{X} = (x_1, \dots, x_n)$ niz promenljivih
- $\mathcal{D} = (D_1, \dots, D_n)$ niz odgovarajućih domena promenljivih, pri čemu je D_i domen promenljive x_i (što ćemo pisati kao $D(x_i) = D_i$)
- $\mathcal{C} = (C_1, \dots, C_k)$ niz ograničenja, pri čemu je svako ograničenje C_i potskup od $D_{i_1} \times \dots \times D_{i_r}$, za neki rastući niz indeksa $i_1 < \dots < i_r$. Kažemo da je C_i ograničenje *nad promenljivama* $x_{i_1}, \dots, x_{i_r}$ (u oznaci $\mathcal{X}(C_i) = (x_{i_1}, \dots, x_{i_r})$). Broj r nazivamo *arnost* ograničenja C

Rešenje CSP problema $\mathcal{P}$ je uređena n -torka $(d_1, \dots, d_n) \in D_1 \times \dots \times D_n$, takva da za svako ograničenje $C_i \in \mathcal{C}$ za koje je $\mathcal{X}(C_i) = (x_{i_1}, \dots, x_{i_r})$, r -torka $(d_{i_1}, \dots, d_{i_r})$ pripada C_i .

Napomena

Ograničenja ćemo obično zapisivati simbolički (npr. $x = y$, $x + y < z$ i sl.). Rešenje $(d_1, \dots, d_n)$ ćemo često zapisivati i kao *dodelu* $\{x_1 = d_1, \dots, x_n = d_n\}$

Napomena

Mi ćemo se ograničiti na CSP probleme nad *konačnim domenima*.

Primer – latinski kvadrat

Primer

Razmotrimo primer *latinskog kvadrata*:

1			
			3
	4		
		2	

Prazna polja u kvadratu treba popuniti brojevima od 1 do 4, tako da u svakoj vrsti i svakoj koloni svi brojevi budu međusobno različiti.

Primer – latinski kvadrat (2)

Primer

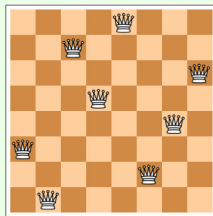
(nastavak) Latinski kvadrat se može predstaviti kao CSP problem na sledeći način:

- *skup promenljivih: x_{ij} za polje u i -toj vrsti i j -toj koloni*
- *domeni svih promenljivih su $\{1, 2, 3, 4\}$*
- *ograničenja su:*
 - $x_{ij_1} \neq x_{ij_2}$ za $1 \leq j_1 < j_2 \leq 4$ i $i \in \{1, 2, 3, 4\}$
 - $x_{i_1j} \neq x_{i_2j}$ za $1 \leq i_1 < i_2 \leq 4$ i $j \in \{1, 2, 3, 4\}$

Primer – 8 kraljica

Primer

*Razmotrimo problem **8 kraljica**: potrebno je rasporediti 8 kraljica na šahovsku tablu tako da se međusobno ne napadaju. Jedno moguće rešenje ovog problema dato je na slici:*



Primer – 8 kraljica (2)

Primer

(nastavak) Problem 8 kraljica se može predstaviti kao CSP problem na sledeći način:

- *imamo promenljive $x_1, \dots, x_8$, sa domenom $\{1, \dots, 8\}$, pri čemu promenljiva x_i označava redni broj kolone u kojoj se nalazi kraljica iz i -te vrste (jasno je da su sve kraljice u različitim vrstama)*
- *imamo ograničenja:*
 - $x_i \neq x_j$ za $i \neq j$ – nikoje dve kraljice se ne smeju nalaziti u istoj koloni
 - za $i \neq j$ mora da važi $|x_i - x_j| \neq |i - j|$, tj. nikoje dve kraljice ne smeju biti na istoj dijagonali

Primer – Problem ranca

Primer

Razmotrimo čuveni *problem ranca* (engl. *knapsack problem*): na raspolaganju nam je ranac fiksiranog kapaciteta K , kao i skup predmeta, pri čemu svaki predmet ima svoju veličinu i svoju vrednost. Zadatak je spakovati što više predmeta u ranac tako da se ne prekorači zadati kapacitet, a da vrednost spakovanih predmeta bude što veća.

NAPOMENA: Umesto ove, optimizacione varijante problema, možemo posmatrati problem odlučivanja: da li postoji rešenje u kome je vrednost spakovanih predmeta veća od neke zadate vrednosti V ?

Primer – Problem ranca (2)

Primer

Problem ranca se može predstaviti kao CSP problem na sledeći način: numerišimo najpre predmete redom sa $1, 2, \dots, n$. Neka je v_i vrednost i -tog predmeta, a s_i njegova veličina. Uvedimo promenljive $x_1, \dots, x_n$ sa domenima $D = \{0, 1\}$. Intuitivno, promenljiva x_i biće jednaka 1 akko je i -ti predmet spakovan u ranac. Sada imamo ograničenje:

$$s_1x_1 + s_2x_2 + \dots + s_nx_n \leq K$$

kojim se obezbeđuje da se kapacitet ranca ne prekorači. Najzad, da bismo obezbedili da vrednost spakovanih predmeta bude bar V , imamo ograničenje:

$$v_1x_1 + v_2x_2 + \dots + v_nx_n \geq V$$

SAT problem kao CSP problem

SAT kao CSP

SAT problem se može razumeti kao poseban tip CSP problema, gde sve promenljive imaju dvočlani domena $\{true, false\}$ i gde su sva ograničenja klauze.

Klauza kao ograničenje

Klauza $C = l_1 \vee l_2 \vee \dots \vee l_k$ gde je $l_i \in \{p_i, \neg p_i\}$, se može razumeti kao ograničenje nad promenljivama $p_1, \dots, p_k$:

$$C = \{true, false\}^k \setminus \{(v_1, \dots, v_k) \mid v_i = false \text{ ako } l_i = p_i, true \text{ inače}\}$$

Opšti CSP rešavači i odnos sa SAT rešavačima

Opšti CSP rešavači

Opšti CSP rešavači su zasnovani na sličnim principima kao i DPLL algoritam:

- **rezonovanje** zasnovano na složenim pravilima propagacije
- **pretraga** zasnovana na podeli domena promenljive i razmatranju odgovarajućih podproblema

Odnos SAT i CSP rešavača

SAT rešavači su po pravilu daleko efikasniji:

- jednostavnost domena i uniformnost ograničenja omogućava efikasne tehnike implementacije
 - u opštem slučaju ne postoje tako efikasne tehnike poput sheme dva posmatrana literala
 - takodje, ne postoje heuristike grananja koje su tako efikasne u opštem slučaju
- Značajna algoritamska poboljšanja takođe nije tako lako realizovati u slučaju opšteg CSP-a:
 - nehnološko vraćanje unazad je korišćeno i u opštem slučaju, ali sa znatno manje uspeha
 - **nogood learning**: tehnika analogna učenju klauza

Zbog toga je veoma čest pristup u novije vreme da se CSP problemi rešavaju svođenjem na SAT.

Svođenje CSP problema na SAT

SAT rešavači i opšti CSP problemi

CSP problemi nad konačnim domenima se mogu svesti na SAT. Najpre treba kodirati domene promenljivih:

- ako je domen promenljive x_i jednak $\{v_1, \dots, v_k\}$, tada možemo uvesti iskazno slovo p_{ij} sa značenjem $x_i = v_j$
 - klauzama $p_{i1} \vee p_{i2} \vee \dots \vee p_{ik}$ i $\neg p_{ij_1} \vee \neg p_{ij_2}$ (za $j_1 < j_2$) obezbeđujemo da tačno jedan od atoma p_{ij} bude tačan, tj. da x_i uzme tačno jednu vrednost iz svog domena
 - opisano kodiranje naziva se **direktno kodiranje** (engl. **direct encoding**)
- alternativno, ako je domen uređen ($v_1 < \dots < v_k$) možemo uvesti iskazna slova q_{ij} sa značenjem $x_i \leq v_j$
 - možemo uvesti klauze: $\neg q_{ij_1} \vee q_{ij_2}$ za $j_1 < j_2$, kao i jediničnu klauzu q_{ik}
 - ovakvo kodiranje se naziva **kodiranje poretka** (engl. **order encoding**)

Nakon toga je potrebno svako ograničenje kodirati skupom klauza (za različite tipove ograničenja postoje različita kodiranja).

Primer – Latinski kvadrat (3)

Primer

(nastavak) Da bismo Latinski kvadrat sveli na SAT, uvedimo iskazna slova p_{ijk} sa značenjem $x_{ij} = k$. Sada imamo klauze:

- $\bigvee_k p_{ijk}$ za svako i, j , kao i $\neg p_{ijk_1} \vee \neg p_{ijk_2}$ za svako i, j i $k_1 < k_2$ (uslovi domena)
- $\neg p_{ij_1k} \vee \neg p_{ij_2k}$ za svako i, k i $j_1 < j_2$ (dva polja u i -toj vrsti ne smeju imati istu vrednost k)
- $\neg p_{i_1jk} \vee \neg p_{i_2jk}$ za svako j, k i $i_1 < i_2$ (dva polja u j -toj vrsti ne smeju imati istu vrednost k)

Primer – 8 kraljica (3)

Primer

(nastavak) Da bismo problem 8 kraljica sveli na SAT, uvodimo promenljive p_{ij} sa značenjem da je i -ta kraljica u j -toj koloni (tj. da je $x_i = j$). Kao i ranije, imaćemo klauze $\bigvee_j p_{ij}$ za svako i , kao i $\neg p_{ij_1} \vee \neg p_{ij_2}$ za $j_1 < j_2$ i svako i . Ove klauze kodiraju uslov da se svaka kraljica nalazi tačno u jednoj koloni. Sada je potrebno obezbediti da za svako j imamo najviše jedno p_{ij} koje je tačno (u svakoj koloni imamo tačno jednu kraljicu). Ovo postićemo klauzama oblika $\neg p_{i_1j} \vee \neg p_{i_2j}$ za $i_1 < i_2$ i svako j . Najzad, za svaki par polja (i_1, j_1) i (i_2, j_2) za koji važi $|i_1 - i_2| = |j_1 - j_2|$ (tj. koja se nalaze na istoj dijagonali) imamo da ne sme biti $x_{i_1} = j_1 \wedge x_{i_2} = j_2$, tj. mora da važi $\neg p_{i_1j_1} \vee \neg p_{i_2j_2}$.

Primer – Ranac (3)

Primer

U problemu ranca, promenljive x_i imaju domene $\{0, 1\}$ pa se već mogu smatrati iskaznim atomima. Pritom, ograničenja koja se u ovom problemu pojavljuju:

$$s_1x_1 + s_2x_2 + \dots + s_nx_n \leq K$$

i

$$v_1x_1 + v_2x_2 + \dots + v_nx_n \geq V$$

*spadaju u posebnu vrstu ograničenja koja nazivamo **pseudo-bulovska ograničenja**, a o kojima govorimo detaljnije u nastavku.*

Pseudo-bulovska ograničenja

Definicija

Ograničenje oblika:

$$c_1 l_1 + c_2 l_2 + \dots + c_n l_n \bowtie d$$

gde su c_i i d celi brojevi, l_i su literali, a $\bowtie \in \{=, <, >, \leq, \geq, \neq\}$ nazivaju se *pseudo-bulovska ograničenja (PB)*. Pritom, prilikom sabiranja vrednost literala l_i računamo kao 1 ako je l_i tačan, a 0 u suprotnom.

Napomena

Specijalni tip ovih ograničenja se dobija kada su svi koeficienti c_i jednaki 1:

$$l_1 + l_2 + \dots + l_n \bowtie d$$

Ovakva ograničenja nazivaju se *ograničenja kardinalnosti*, jer se njima ograničava broj literala $l_1, \dots, l_n$ koji su tačni.

Napomena

Primetimo da se svaka klauza:

$$l_1 \vee \dots \vee l_n$$

može smatrati ograničenjem kardinalnosti:

$$l_1 + \dots + l_n \geq 1$$

Otuda se problem zadovoljenja ograničenja kardinalnosti (pa i pseudo-bulovskih ograničenja) može smatrati uopštenjem KNF SAT problema.

Primer

Primer

Setimo se da smo u latinskom kvadratu zahtevali da među promenljivama p_{ijk} za fiksirano i i k najviše jedna bude tačna (tj. da najviše jedno polje u i -toj vrsti ima vrednost k). Ovo se može jednostavno predstaviti ograničenjem kardinalnosti:

$$p_{i1k} + p_{i2k} + p_{i3k} + p_{i4k} \leq 1$$

Ovo ograničenje se jednostavno kodiralo skupom klauza $\neg p_{ij_1k} \vee \neg p_{ij_2k}$ za svako $j_1 < j_2$. Međutim, da li je tako jednostavno kodirati ovakva ograničenja u opštem slučaju (npr. kada je desna strana veća od 1)?

Pseudo-bulovska ograničenja

Kodiranje pseudo-bulovskih ograničenja na jeziku iskazne logike

- Direktno kodiranje pseudo-bulovskih ograničenja pomoću klauza može zahtevati eksponencijalno mnogo klauza
- Postoje efikasnija kodiranja koja podrazumevaju uvođenje dodatnih iskaznih atoma (slično kao kod Cajtina)
- Jedno kodiranje se može dobiti simulacijom binarnih sabirača:
 - razmatramo ograničenje $c_1 l_1 + c_2 l_2 + \dots + c_n l_n \geq d$, gde je $c_i \geq 0$ i $d \geq 0$ (svaki PB problem možemo svesti na ovakva ograničenja)
 - neka je $S_2 = c_1 l_1 + c_2 l_2$
 - zbir S_2 se predstavi binarnim sabiračem, gde se u binarnim zapisima brojeva c_1 i c_2 jedinice zamenjuju literalima l_1 i l_2 , respektivno
 - u binarnom sabiraču se uvode dodatne promenljive koje označavaju prenose između pozicija, a onda se vrši prevođenje u KNF
 - nakon toga se uvodi $S_3 = S_2 + c_3 l_3$ i predstavlja se sabiračem na sličan način, i td. ($S_i = S_{i-1} + c_i l_i$)
 - na kraju se realizuje binarni komparator koji ispituje da li je $S_n \geq d$ i on se prevodi u KNF

Optimizacioni PB problemi

Optimizacija PB izraza

Često je potrebno rešiti pseudo-bulovski problem, tako da rešenje maksimizuje (ili minimizuje) dati pseudo-bulovski izraz: $c_1 l_1 + \dots c_n l_n$:

- pretpostavimo da je u pitanju minimizacija i da smo pronašli rešenje za koje je vrednost datog izraza K
- dodajemo ograničenje $c_1 l_1 + \dots c_n l_n < K$ i ponovo rešavamo problem
- ako je sada problem nezadovoljiv, tada je K upravo minimalna vrednost izraza $c_1 l_1 + \dots + c_n l_n$
- u suprotnom neka je vrednost datog izraza u pronađenom rešenju K' (koje je sigurno manje od K)
- sada dodajemo ograničenje $c_1 l_1 + \dots c_n l_n < K'$ i ponovo rešavamo problem, i td.

Opisana strategija se naziva **linearna strategija**. Alternativa je tzv. **binarna strategija**, gde se optimalna vrednost traži binarnom pretragom:

- u svakom trenutku imamo neki interval $[L, U]$ u kome tražimo minimalnu vrednost funkcije
- za K uzimamo sredinu intervala $(L + U)/2$ i dodajemo ograničenje $c_1 l_1 + \dots + c_n l_n < K$
- ako je problem zadovoljiv, tada pretragu nastavljamo u intervalu $[L, K]$, a ako je nezadovoljiv, nastavljamo u $[K, U]$

U praksi se obično koristi linearna strategija, jer je efikasnija:

- iako binarna strategija konvergira brže, ona zahteva rešavanje većeg broja nezadovoljivih problema
- sa druge strane, linearna strategija podrazumeva veći broj zadovoljivih i jedan nezadovoljiv problem
- nezadovoljivi problemi obično zahtevaju više vremena za rešavanje
- otuda linearna strategija obično zahteva manje vremena za pronalaženje optimalnog rešenja

MaxSAT problem

Definicija

*Za datu KNF formulu, problem **MaxSAT** se sastoji u pronalaženju valuacije u kojoj je zadovoljen najveći mogući broj klausa.*

Napomena

MaxSAT problem se može razumeti kao uopštenje SAT problema: KNF formula je zadovoljiva akko je najveći mogući broj zadovoljenih klausa u nekoj valuaciji jednak ukupnom broju klausa.

Definicija

*Uopštenje MaxSAT problema je **težinski MaxSAT**: svakoj klauzi C_i se dodeljuje težinski koeficient $w_i \geq 0$. Sada je cilj da se pronade valuacija takva da je zbir težina klausa koje su tačne u toj valuaciji maksimalan.*

MaxSAT problem

Svođenje težinskog MaxSAT problema na PB problem

- Neka je dat skup klauza $\{C_1, \dots, C_m\}$, pri čemu klauza C_i ima težinu w_i
- Za svaku klauzu $C_i \equiv l_{i1} \vee l_{i2} \vee \dots \vee l_{in_i}$ uvodimo novo iskazno slovo r_i
- Klauzu C_i zamenjujemo uslovom $r_i \Rightarrow C_i$, tj. klauzom $\neg r_i \vee l_{i1} \vee \dots \vee l_{in_i}$
- Sada je cilj maksimizovati PB izraz $w_1 r_1 + w_2 r_2 + \dots + w_m r_m$

Linearna ograničenja

Linearna ograničenja

Daljim uopštenjem pseudo-bulovskih ograničenja dolazimo do **linearnog ograničenja**:

$$a_1x_1 + a_2x_2 + \dots + a_nx_n \bowtie b$$

gde su a_i i b celobrojne konstante, x_i su promenljive čiji su domeni konačni celobrojni intervali $[l_i, u_i]$, a $\bowtie \in \{=, \neq, \leq, \geq, <, >\}$.

Napomena

Lako se može pokazati da se CSP koji uključuje linearna ograničenja uvek može transformisati u ekvivalentan CSP u kome sva linearna ograničenja imaju oblik:

$$a_1x_1 + a_2x_2 + \dots + a_nx_n \leq b$$

Primer – magični kvadrat

Primer

Dat je *magični kvadrat*:

	12		
	8	15	
7		2	
4			11

Potrebno je prazna polja popuniti celim brojevima iz intervala $[1, 16]$, tako da svi brojevi u tablici budu različiti, a zbir po svim vrstama, kolonama i velikim dijagonalama bude 34.

Primer – magični kvadrat (2)

Primer

(nastavak) Magični kvadrat se jednostavno predstavlja kao CSP problem:

- *Uvodimo promenljive x_{ij} za polje u i -toj vrsti i j -toj koloni ($1 \leq i, j \leq 4$)*
- *Sve promenljive imaju domen $\{1, 2, \dots, 16\}$*
- *Skup ograničenja:*
 - $x_{i1} + \dots + x_{i4} = 34$, za svako $i \in \{1, 2, 3, 4\}$
 - $x_{1j} + \dots + x_{4j} = 34$, za svako $j \in \{1, 2, 3, 4\}$
 - $x_{11} + x_{22} + x_{33} + x_{44} = 34$
 - $x_{41} + x_{32} + x_{23} + x_{14} = 34$
 - $x_{12} = 12, x_{22} = 8, x_{23} = 15, x_{31} = 7, x_{33} = 2, x_{41} = 4, x_{44} = 11$
 - $x_{ij} \neq x_{i'j'}$ za svaka dva različita polja (i, j) i (i', j')

Linearna ograničenja

Svođenje na SAT

Neka je dato ograničenje:

$$a_1x_1 + a_2x_2 + \dots + a_nx_n \leq b$$

Jednostavnosti radi, pretpostavimo da su svi a_i nenegativni. Ako je $x_i \geq c_i$ (za $i = 1, 2, \dots, n-1$), tada mora biti $x_n \leq c_n$, gde je

$$c_n = \left\lfloor \left(b - \sum_{1 \leq i < n} a_i c_i \right) / a_n \right\rfloor$$

Otuda imamo implikaciju $x_1 \geq c_1 \wedge x_2 \geq c_2 \wedge \dots \wedge x_{n-1} \geq c_{n-1} \Rightarrow x_n \leq c_n$, koja je ekvivalentna sa: $x_1 \leq c_1 - 1 \vee x_2 \leq c_2 - 1 \vee \dots \vee x_{n-1} \leq c_{n-1} - 1 \vee x_n \leq c_n$. Ako uvedemo iskazno slovo q_i^k sa značenjem $x_i \leq k$ ($k \in [l_i, u_i]$), sada se ovaj uslov može zapisati kao iskazna klauza (dakle, koristimo [kodiranje poretka](#) za predstavljanje domena). Ovakve klauze treba napraviti za sve moguće vrednosti c_i iz domena odgovarajućih promenljivih.

Linearna ograničenja

Svođenje na SAT

(nastavak) Ukupan broj klauza dobijen na prethodni način je $O(d^{n-1})$, gde je d veličina najvećeg domena, a n broj promenljivih u linearnom ograničenju. Ovaj broj se može značajno smanjiti, ako se najpre linearno ograničenje transformiše na sledeći način:

- uvedemo promenljive $y_2 = a_1x_1 + a_2x_2$, $y_3 = y_2 + a_3x_3$, ..., $y_{n-2} = y_{n-3} + a_{n-2}x_{n-2}$
- ograničenje $y_i = y_{i-1} + a_ix_i$ možemo zameniti parom ograničenja $y_{i-1} + a_ix_i - y_i \leq 0$ i $-y_{i-1} - a_ix_i + y_i \leq 0$
- dodatno, imamo ograničenje $y_{n-2} + c_{n-1}x_{n-1} + c_nx_n \leq b$

Dakle, sveli smo ovo ograničenje na konjunkciju linearnih ograničenja sa po tri promenljive. Sada svako od tih ograničenja predstavimo klauzama na prethodno opisan način. Broj klauza će sada biti $O(nd^2)$, što je obično značajno manje.

Optimizacioni problemi sa linearnim ograničenjima

Primerdba

Optimizacioni problemi kod kojih je potrebno minimizovati (maksimizovati) linearni celobrojni izraz $a_1x_1 + \dots + a_nx_n$ se mogu rešavati na sličan način kao i optimizacioni pseudo-bulovski problemi (korišćenjem linearne ili binarne strategije).