UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Matija Lojović

SVOĐENJE PROBLEMA ZADOVOLJENJA OGRANIČENJA ZADATIH NA JEZIKU FLATZINC NA PROBLEM SAT

master rad

Beograd, 2025.

Mentor:

dr Milan Banković, docent
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Filip Marić, redovni profesor
Univerzitet u Beogradu, Matematički fakultet

dr Vesna Marinković, vanredni profesor
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Naslov master rada: Svođenje problema zadovoljenja ograničenja zadatih na jeziku FlatZinc na problem SAT

Rezime: U ovom radu je predstavljen alat za svođenje problema zadovoljenja ograničenja na problem SAT. Pored neophodnih teorijskih osnova, dat je i opis ulaznog jezika alata pod nazivom FlatZinc. Procedura kodiranja FlatZinc promenljivih, kao i pojedinačnih FlatZinc ograničenja celobrojnog, logičkog i skupovnog tipa detaljno je opisana i analizirana. Ključni detalji implementacije, kao i detaljna evaluacija alata izloženi su u poslednje dve glave ovog rada.

Ključne reči: problem CSP, problem SAT, FlatZinc, konverzija CSP u SAT

Sadržaj

Glava 1

Uvod

Pojam problema zadovoljenja ograničenja (engl. *Constraint satisfaction problem* (*problem CSP*)) pojavio se sedamdesetih godina prošlog veka [7]. Vrlo brzo je dobio na popularnosti zbog mogućnosti predstavljanja brojnih praktičnih problema u obliku problema CSP. Lista domena u kojima se problem CSP pojavljuje uključuje obradu prirodnih jezika [4], planiranje i alokaciju resursa [2], automatsko dokazivanje teorema [5] i mnoge druge.

U svojoj osnovnoj formi, problem CSP podrazumeva dodelu vrednosti svakoj od promenljivih koje učestvuju u problemu. Svaka od promenljivih uzima vrednost iz svog fiksiranog domena, dok ograničenja koja figurišu u problemu određuju koje vrednosti promenljivih ne mogu ići zajedno. U slučaju da postoji dodela vrednosti promenljivama takva da poštuje sva ograničenja, kažemo da problem ima rešenje. U suprotnom, problem nema rešenja.

Razvijeni su brojni alati u cilju rešavanja problema CSP pod nazivom *CSP rešavači*. Oni su najčešće zasnovani ne nekoj vrsti pretrage sa vraćanjem (engl. *backtracking search*) ili lokalne pretrage. Takođe, obično se koriste tehnike pojednostavljenja problema pre primene pretrage, kolektivno poznate kao propagacija ograničenja.

Sa razvojem CSP rešavača pojavila se potreba za standardizovanim jezicima za modelovanje problema CSP. Jedan od trenutno najpopularnijih takvih jezika nosi naziv *MiniZinc* [15]. Kako je MiniZinc vrlo izražajan jezik, modeli zapisani u njemu mogu biti izuzetno kompleksni. Iz tog razloga, pre nego što bude predat CSP rešavaču, ulaz na jeziku MiniZinc se najčešće prevodi u njegovu pojednostavljenu formu, poznatu kao *FlatZinc* [14].

Problem ispitivanja zadovoljivosti iskazne formule (engl. *Boolean satisfiability problem* (*problem SAT*)) jedan je od centralnih problema teorijskog računarstva još

od sedamdesetih godina prošlog veka [1]. Tada je dokazano da je problem SAT NP-kompletan, postavši tako prvi problem za koji je formalno dokazana pripadnost ovoj klasi složenosti. U međuvremenu, razvijen je veliki broj izuzetno efikasnih alata za rešavanje ovog problema pod nazivom *SAT rešavači*. Oni najčešće na ulazu očekuju formulu u *konjunktivnoj normalnoj formi* (*KNF*).

Jedan od mogućih načina rešavanja problema CSP uključuje njegovo svođenje na problem SAT [9]. Dva glavna pristupa tome su vredno (engl. *eager*) i lenjo generisanje klauza (engl. *lazy clause generation* [11]). Ključna razlika između pristupa se ogleda u tome što se kod vrednog generisanja klauza čitav model unapred prevodi u KNF i zatim predaje postojećem SAT rešavaču na rešavanje, dok se kod lenjog pristupa SAT rešavač koristi interno u okviru CSP rešavača za pretragu koja se usmerava dinamičkim generisanjem klauza koje kodiraju ograničenja u toku pretrage. Postoje alati zasnovani, kako na prvom (Sugar [17]), tako i na drugom pristupu (Chuffed [13], geas [16]). Dok alati zasnovani na lenjom generisanju klauza uglavnom podržavaju FlatZinc kao ulazni jezik, alati zasnovani na vrednom pristupu po pravilu koriste svoje specifične ulazne jezike. Koliko je autoru ovog rada poznato, ne postoji alat koji problem CSP predstavljen na jeziku FlatZinc konvertuje u KNF.

Cilj ovog rada biće implementacija i predstavljanje jednog takvog alata, zasnovanog na vrednom pristupu generisanja klauza. Alat će vršiti kodiranje FlatZinc promenljivih, kao i ograničenja celobrojnog, logičkog i skupovnog tipa u *DIMACS format* [3] koji podržava većina modernih SAT rešavača. Pored toga, biće omogućeno dekodiranje izlaza SAT rešavača u slučaju zadovoljivih problema.

Ostatak rada je organizovan na sledeći način. U glavi 2 biće predstavljene neophodne teorijske osnove, zajedno sa ilustrativnim primerima. U glavi 3 biće opisan i analiziran način kodiranja FlatZinc promenljivih i pojedinačnih ograničenja. U glavi 4 biće dat osvrt na ključne implementacione detalje alata. U glavi 5 biće predstavljena detaljna evaluacija performansi alata. Glava 6, kao zaključak, predočiće postignute rezultate ovog rada, kao i predloge za potencijalni dalji rad.

Glava 2

Osnove

2.1 Problem zadovoljenja ograničenja

Problem zadovoljenja ograničenja (problem CSP) P je uređena trojka $P = (X, D, C)$, pri čemu važi:

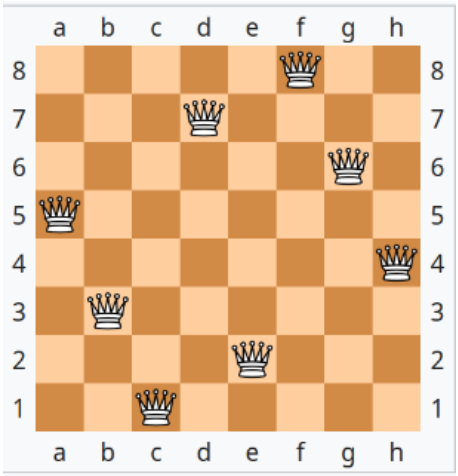
- $X = (x_1, \ldots, x_n)$ je niz promenljivih
- $D = (D_1, \ldots, D_n)$ je niz odgovarajućih domena promenljivih, pri čemu je D_i domen promenljive x_i (skraćeno $D(x_i) = D_i$)
- $C = (C_1, \ldots, C_m)$ je niz ograničenja, pri čemu je svako ograničenje C_i podskup $D_{i_1} \times \cdots \times D_{i_k}$, za neki rastući niz indeksa $i_1, \ldots, i_k$. Kažemo da promenljive $x_{i_1}, \ldots, x_{i_k}$ figurišu u ograničenju C_i (skraćeno $X(C_i) = (x_{i_1}, \ldots, x_{i_k})$), a broj k nazivamo arnost ograničenja.

Rešenje problema CSP P je uređena n-torka $(d_1, \ldots, d_n) \subseteq D_1 \times \cdots \times D_n$, takva da za svako ograničenje C_i nad promenljivama $(x_{i_1}, \ldots, x_{i_k})$, k-torka $d_{i_1}, \ldots, d_{i_k}$ pripada C_i. Najčešće se rešenje problema CSP $(d_1, \ldots, d_n)$ zapisuje kao dodela vrednosti promenljivama $\{x_1 = d_1, \ldots, x_n = d_n\}$. Takođe, najčešće se pod rešavanjem problema CSP podrazumeva pronalaženje bilo kog rešenja, a ređe, pronalaženje svih mogućih rešenja.

Domen CSP promenljive može biti konačan, prebrojivo beskonačan ili neprebrojivo beskonačan. U ovom radu, pažnja će biti posvećena problemima CSP sa konačnim domenima promenljivih.

Primer 1. *Primer problema koji može biti modelovan kao problem CSP sa konačnim domenima promenljivih je* problem osam dama. *Cilj ovog problema je postaviti osam*

dama na šahovsku tablu tako da se međusobno ne napadaju. Jedno moguće rešenje, kao i primer modelovanja problema osam dama u obliku problema CSP dati su u nastavku:



Slika 2.1: Jedno moguće rešenje problema osam dama.[1]

- ***Promenljive:*** x_i *za* $i \in \{1, \ldots, 8\}$*, gde svako* x_i *predstavlja redni broj vrste u kojoj će biti postavljena dama iz* i*-te kolone.*
- ***Domeni:*** $D_i = \{1, \ldots, 8\}$ *za svaku promenljivu* x_i*.*
- ***Ograničenja:***
  1. ***Ograničenja vrsta:*** *Svaka vrsta mora sadržati tačno jednu damu:*

$$\forall i, j \in \{1, \ldots, 8\} \quad i < j \implies x_i \neq x_j$$

  2. ***Ograničenja dijagonala:*** *Dve dame ne smeju biti postavljene na istoj dijagonali:*

$$\forall i, j \in \{1, \ldots, 8\} \quad i < j \implies |x_i - x_j| \neq |i - j|$$

Jedno rešenje ovog problema je $\{x_1 = 5, x_2 = 3, x_3 = 1, x_4 = 7, x_5 = 2, x_6 = 8, x_7 = 6, x_8 = 4\}$ *(slika 2.1).*

[1]Slika preuzeta sa `https://en.wikipedia.org/wiki/Eight_queens_puzzle`

2.2 MiniZinc i FlatZinc

MiniZinc je jezik visokog nivoa za modelovanje problema CSP [10]. Sadrži veliku biblioteku predefinisanih ograničenja koja mu omogućava izuzetnu ekspresivnost pri modelovanju. Takođe, MiniZinc podržava različite mehanizme koji olakšavaju proces modelovanja, poput struktura podataka visokog nivoa (nizovi, skupovi, enumerisani tipovi), korisnički definisanih funkcija i predikata, kao i razgraničenja podataka i modela.

Primer gorepomenutog problema osam dama modelovanog pomoću jezika MiniZinc dat je u nastavku:

```
1
2  par int : n = 8;
3
4  array[1..n] of var 1..n: queens;
5
6  constraint
7    forall(i in 1..n) (
8      forall(j in i+1..n) (
9        queens[i] != queens[j] /\
10       abs(queens[i] - queens[j]) != abs(i - j)
11     )
12   );
13
14 solve satisfy;
15
16 output [ show(queens) ];
```

U navedenom modelu, po ključnoj reči *par* može se prepoznati da n predstavlja MiniZinc parametar - simboličku konstantu čija je vrednost poznata u fazi prevođenja u FlatZinc, a čija se vrednost može zadati i u zasebnoj datoteci. Ključna reč *var* koristi se pri deklaraciji MiniZinc promenljive čiji se domen zadaje, a od rešavača se očekuje da joj dodeli vrednost. Ključna reč *array* govori da je u pitanju deklaracija niza, a konstrukti $[1..n]$ i $1..n$ određuju redom njegovu dimenziju (niz ima n elemenata), odnosno domen svakog od elemenata (skup celih brojeva $\{1, \ldots, n\}$). U kombinaciji, ključne reči *array* i *var* govore da je u pitanju deklaracija nizovske promenljive - niza čiji su elementi promenljive. Nakon ključne reči *constraint* navodi se ograničenje, a nakon ključne reči *solve* cilj modela (zadovoljenje ili optimizacija).

FlatZinc je podskup jezika MiniZinc čija je primarna uloga da služi kao ulaz za CSP rešavače. Njegova sintaksa je znatno jednostavnija (u smislu ograničenja

koja se mogu zadati) od sintakse jezika MiniZinc, ali za razliku od MiniZinc modela, FlatZinc modeli često umeju da budu glomazni i nečitljivi za čoveka. U okviru standardne MiniZinc distribucije[2] postoji alat koji automatski prevodi ulaz zadat na jeziku MiniZinc u FlatZinc reprezentaciju koja se dalje može proslediti odgovarajućem rešavaču.

Kao i MiniZinc modeli, FlatZinc modeli se sastoje iz:

- nula ili više spoljašnjih deklaracija predikata (nestandardni predikati podržani od strane konkretnih rešavača)
- nula ili više deklaracija parametara
- nula ili više deklaracija promenljivih
- nula ili više ograničenja
- jednog cilja modela (zadovoljenje, maksimizacija, minimizacija)

Ključna razlika u odnosu na MiniZinc modele je u tome što su ograničenja koja se mogu zadavati u FlatZinc modelu znatno jednostavnija. FlatZinc ograničenja mogu se svrstati u jedan od četiri tipa: celobrojna, logička, skupovna ili realna. Slično, promenljive u jeziku FlatZinc mogu biti celobrojnog, logičkog, skupovnog, realnog, ali i nizovskog tipa, tj. mogu predstavljati niz vrednosti nekog od osnovnih tipova. Parametri su fiksirane vrednosti koje figurišu u modelu i mogu biti istih tipova kao i promenljive. Cilj ovog rada je prevođenje FlatZinc modela koji predstavljaju probleme CSP, pa će se podrazumevati da je cilj modela njegovo zadovoljenje, a ne optimizacija određenog izraza. Takođe, biće obrađena standardna FlatZinc ograničenja, pa će odsustvo spoljašnjih predikata biti podrazumevano.

Prevođenje MiniZinc modela u FlatZinc model naziva se *poravnanje* (engl. *flattening*). Poravnanjem se najčešće uvode pomoćne promenljive koje nisu postojale u osnovnom MiniZinc modelu, a složena MiniZinc ograničenja se raščlanjavaju na jednostavnija FlatZinc ograničenja.

Poravnanjem MiniZinc modela osam dama uvodi se osam novih promenljivih koje predstavljaju elemente niza *queens*. Ograničenja vrsta svode se na ograničenja oblika $int\_lin\_ne([-1,1],[x_i,x_j],0)$ čime se obezbeđuje da linearna jednakost $-x_i + x_j = 0$ ne bude tačna. Za ograničenja dijagonala najpre se uvode pomoćne

[2] `https://www.minizinc.org`

promenljive x_{pom} koje će čuvati razlike oblika $x_i - x_j$. Njihove vrednosti se postavljaju korišćenjem ograničenja oblika $int\_lin\_eq([1, -1, -1], [x_i, x_j, x_{pom}], 0)$, čime se obezbeđuje da linearna jednačina $x_i - x_j - x_{pom} = 0$ bude ispunjena. Dalje, uvode se pomoćne promenljive x_{bez_k} čiji su domeni oblika $\{0, \ldots, 7\} \setminus k, k \in \{0, \ldots, 7\}$. One efektivno pamte za par promenljivih x_i, x_j razliku njihovih indeksa $i - j$ kao broj k koji im nedostaje u domenu. Ograničenja oblika $int\_abs(x_{pom}, x_{bez\_k})$ se staraju da razlika $x_i - x_j$ sačuvana u promenljivoj x_{pom} ne bude jednaka k, tj. $i - j$, a samim tim i obezbeđuju ograničenja dijagonala. Više reči o pomenutim FlatZinc ograničenjima dato je u glavi 3.

2.3 Problem SAT

Neka je dat najviše prebrojiv skup iskaznih promenljivih (atoma) P i skup logičkih veznika $\{\bot, \top, \neg, \wedge, \vee, \Rightarrow, \Leftrightarrow\}$. Skup iskaznih formula F_P nad P formira se na sledeći način:

- atomi i logičke konstante $(\bot, \top)$ su iskazne formule
- ako je A iskazna formula, onda je i $\neg A$ iskazna formula
- ako su A i B iskazne formule, onda su i $A \wedge B$, $A \vee B$, $A \Rightarrow B$, $A \Leftrightarrow B$ iskazne formule
- ako je A iskazna formula, onda je i (A) iskazna formula

Valuacija $v : P \to \{0, 1\}$ predstavlja funkciju koja dodeljuje atomima njihovu istinitosnu vrednost. Interpretacija $I_v : F_P \to \{0, 1\}$ je funkcija indukovana valuacijom v koja svakoj formuli iz skupa iskaznih formula F_P dodeljuje istinitosnu vrednost. Interpretacija se definiše rekurzivno:

- $I_v(p) = 1$ akko $v(p) = 1$, gde je p atom
- $I_v(\top) = 1, I_v(\bot) = 0$
- $I_v(\neg A) = 1$ akko $I_v(A) = 0$
- $I_v(A \wedge B) = 1$ akko $I_v(A) = 1$ i $I_v(B) = 1$
- $I_v(A \vee B) = 1$ akko $I_v(A) = 1$ ili $I_v(B) = 1$

- $I_v(A \Rightarrow B) = 1$ akko $I_v(A) = 0$ ili $I_v(B) = 1$
- $I_v(A \Leftrightarrow B) = 1$ akko $I_v(A) = I_v(B)$

Iskazna formula F je zadovoljiva ukoliko postoji valuacija v u kojoj se interpretira kao tačna, odnosno ukoliko za neku valuaciju v važi $I_v(F) = 1$. Ovo označavamo sa $v \models F$ i kažemo da je v model za F.

Problem SAT predstavlja problem ispitivanja zadovoljivosti iskazne formule u proizvoljnom obliku[3]. Najčešće se proučava specijalni slučaj ovog problema nad formulama u konjunktivnoj normalnoj formi (KNF). Iskazna formula F je u KNF ukoliko je oblika $K_1 \wedge \cdots \wedge K_n$, gde su $K_i, i \in \{1 \ldots n\}$ klauze, odnosno iskazne formule oblika $l_1 \vee \cdots \vee l_m$, pri čemu su $l_j, j \in \{1 \ldots m\}$ literali (atomi ili njihove negacije).

Problem SAT pripada klasi NP kompletnih problema [1]. Bitno je napomenuti da ima veliki broj praktičnih primena, pošto se mnogi problemi u praksi mogu svesti na problem SAT. Većina modernih SAT rešavača zasnovana je na *CDCL algoritmu* [8], što im omogućava da efikasno rešavaju probleme sa više hiljada promenljivih i više desetina hiljada klauza.

2.4 Svođenje problema CSP na problem SAT

Svođenje problema CSP na problem SAT pojavilo se kao ideja krajem devedesetih godina prošlog veka [18], a glavna motivacija je bio nagli napredak u efikasnosti SAT rešavača. Prvi pristup koji se pojavio uključuje kodiranje kompletnog problema CSP kao iskazne formule u KNF obliku, koja se zatim predaje SAT rešavaču na rešavanje. Postoji više vrsta kodiranja:

- **retko** - svaka vrednost iz domena se kodira kao jedna iskazna promenljiva
- **direktno** - varijanta retkog kodiranja; za svaku nedozvoljenu kombinaciju vrednosti promenljivih dodaje se po jedna klauza
- **potporno** - varijanta retkog kodiranja; koristi klauze da predstavi ograničenja oblika „ako a ima vrednost i, onda b mora imati neku od vrednosti iz skupa I"

[3]Ponekad se u literaturi termin „SAT problem" odnosi isključivo na problem ispitivanja zadovoljivosti iskazne formule u KNF obliku, dok se ponekad ta varijanta problema eksplicitno označava kao CNF-SAT. Iako većina SAT rešavača očekuje na ulazu isključivo KNF formulu, postoje i „neklauzalni" SAT rešavači koji mogu na ulazu dobiti iskaznu formulu u proizvoljnom obliku.

- **uređeno** - za svaku vrednost a iz domena promenljive x postoji iskazna promenljiva koja je tačna akko $x \leq a$
- **log (binarno)** - iskazne promenljive kodiraju cifre u binarnom zapisu broja

Takođe, postoji mogućnost da se kodiranja kombinuju u okviru jednog problema CSP, u zavisnosti od konkretnih ograničenja. Opisani pristup naziva se *vredno generisanje klauza* (engl. *eager encoding*) i alat izložen u ovom radu počiva upravo na tom pristupu.

Alternativni pristup svođenju problema CSP na problem SAT je *lenjo generisanje klauza* (engl. *lazy clause generation* (LCG)) [11]. Ovaj pristup kombinuje vrline SAT i CSP rešavača - koriste se mehanizmi učenja konfliktnih klauza (engl. *nogood learning*) i povratnih skokova (engl. *backjumping*) iz prvih, kao i propagacija ograničenja iz drugih. Rezultat je alat koji u osnovi ima CSP rešavač, a u pozadini SAT rešavač za koji se klauze generišu prilikom propagacije ili konflikta.

Glava 3

Svođenje FlatZinc ograničenja na problem SAT

3.1 Kodiranje domena promenljivih

Kako je centralna tema ovog rada svođenje FlatZinc ograničenja celobrojnog, logičkog i skupovnog tipa na problem SAT, najpre je neophodno kodirati domene promenljivih pomenutih tipova na jezik koji SAT rešavači razumeju - u obliku KNF formula. Preciznije, svaka promenljiva će, u zavisnosti od svog domena, doprineti određenim klauzama finalnoj KNF formuli koja predstavlja problem CSP. Takođe, potrebno je formirati preslikavanje koje jednoznačno određuje vrednost promenljive na osnovu zadovoljavajuće valuacije KNF formule. Iskazne promenljive korišćene za kodiranje domena CSP promenljivih kasnije će figurisati u kodiranju ograničenja nad tim promenljivama. Treba napomenuti da se nizovske promenljive kodiraju tako što se svaki element niza kodira kao posebna promenljiva odgovarajućeg tipa (u daljem tekstu označene sa $x[i]$, gde je x ime niza, a i indeks promenljive u nizu).

Celobrojne promenljive

Celobrojne promenljive primarno su kodirane pomoću uređenog kodiranja. Kao što je ranije pomenuto, ono podrazumeva uvođenje iskazne promenljive $p_{x,a}$ za svaku vrednost a iz domena promenljive x koja je tačna akko $x \leq a$. Formalno,

$$\forall a \in \{l(x) - 1, \ldots, u(x)\}, \quad p_{x,a} \Leftrightarrow x \leq a$$

pri čemu su $l(x)$ i $u(x)$ donja, odnosno gornja granica domena promenljive x. Takođe, kako bi se obezbedila konzistentnost dodele (svakoj promenljivoj je dodeljena tačno jedna vrednost iz domena), dodaju se sledeće klauze u KNF formulu:

$$\forall i \in \{l(x), \ldots, u(x)\}, \quad \neg p_{x,i-1} \vee p_{x,i}$$

kao i sledeće jedinične granične klauze:

$$\neg p_{x,l(x)-1}$$

$$p_{x,u(x)}$$

Vrednost ovako kodirane promenljive x može se odrediti pronalaženjem vrednosti $i \in D(x)$ za koju važi da je $p_{x,i}$ tačno, a $p_{x,i-1}$ netačno u dobijenoj iskaznoj valuaciji koja zadovoljava KNF formulu (ukoliko je problem zadovoljiv, ovakva vrednost sigurno postoji i jedinstvena je zbog uslova konzistentnosti dodele).

U slučaju da je domen promenljive oblika intervala $[i, j]$, ovo je dovoljno za njegovo kodiranje. Ukoliko to nije slučaj, domen se može posmatrati kao interval koji sadrži „rupe", tj. $[i, j] \setminus ([k_1, l_1] \cup \cdots \cup [k_n, l_n])$. Kako bi se sprečilo da promenljiva uzme neku od nedozvoljenih vrednosti, potrebno je dodati sledeće klauze:

$$\forall i \in \{1 \ldots n\}, \quad p_{x,k_i-1} \vee \neg p_{x,l_i}$$

Pored uređenog kodiranja, za kodiranje celobrojnih domena je u nekoliko situacija korišćeno i retko kodiranje. Kod retkog kodiranja, svaka vrednost iz domena promenljive predstavljena je pomoću jedne iskazne promenljive:

$$\forall a \in D(x), \quad s_{x,a} \Leftrightarrow x = a$$

Kako je uređeno kodiranje primarno i primenjuje se za svaku celobrojnu promenljivu, potrebno je dodati klauze koje ostvaruju vezu između ove dve vrste kodiranja:

$$\forall a \in D(x), \quad s_{x,a} \Leftrightarrow p_{x,a} \wedge \neg p_{x,a-1}$$

odnosno nakon svođenja na KNF:

$$(\neg s_{x,a} \vee p_{x,a}) \wedge (\neg s_{x,a} \vee \neg p_{x,a-1}) \wedge (s_{x,a} \vee \neg p_{x,a} \vee p_{x,a-1})$$

Bitno je napomenuti da, u slučaju retkog kodiranja, nije potrebno dodavati klauze koje obezbeđuju konzistentnost dodele, pošto je to implicitno urađeno vezom sa uređenim kodiranjem.

Logičke promenljive

Svaka logička promenljiva može se jednostavno kodirati pomoću jedne iskazne promenljive:

$$q_x \Leftrightarrow x = true$$

Skupovne promenljive

U jeziku FlatZinc, domen skupovne promenljive x je partitivni skup nekog datog skupa A_x (tj. $D(x) = \mathcal{P}(A_x)$). Skupovne promenljive kodirane su pomoću niza iskaznih promenljivih, pri čemu iskazna promenljiva $r_{x,a}$ govori da li je element a iz skupa A_x uključen u vrednost promenljive x ili ne:

$$\forall a \in A_x, \quad r_{x,a} \Leftrightarrow a \in x$$

Može se primetiti da u slučaju valuacije koja dodeljuje svim $r_{x,a}$ za $a \in D(x)$ vrednost netačno, x postaje prazan skup.

3.2 Pomoćni koncepti

Pre samog kodiranja ograničenja, u ovom odeljku biće izloženo nekoliko pomoćnih koncepata koji olakšavaju dalji proces kodiranja. Nakon toga, biće prikazan sam proces kodiranja pojedinačnih ograničenja, podeljenih po tipu. Težiće se da slična ograničenja budu grupisana, kao i da se u slučaju kada kodiranje jednog ograničenja može poslužiti kao osnov za kodiranje drugog, uvek prvo bude izloženo jednostavnije ograničenje. Takođe, biće izložen primer koji ilustruje kodiranje različitih ograničenja na konkretnom FlatZinc modelu.

Kodiranje primitivnog zbira/razlike

Kao što je detaljnije objašnjeno u literaturi [12], kodiranje primitivnog zbira oblika $x + y \leq c$, pri čemu su x i y celobrojne promenljive, a c celobrojna konstanta, može se izvršiti uređenim kodiranjem na sledeći način:

$$\bigwedge_{a+b=c-1} (p_{x,a} \vee p_{y,b})$$

Parametri a i b su celobrojni i važi $a \in \{l(x) - 1, \ldots, u(x)\}$, $b \in \{l(y) - 1, \ldots, u(y)\}$. Slična je situacija kada je u pitanju primitivna razlika oblika $x - y \leq c$, pri čemu za parametar b važi $b \in \{-u(y) - 1, \ldots, -l(y)\}$, a umesto literala $p_{y,b}$, dodaje se literal $\neg p_{y,-b-1}$.

Dodatno, u oba slučaja pre kodiranja treba povesti računa da li su granični uslovi zadovoljeni, tj. da li važi $c \geq l(x) + l(y)$ u slučaju zbira, odnosno $c \geq l(x) - r(y)$ u slučaju razlike. Ukoliko to nije slučaj, nejednakost ne može biti zadovoljena.

Korišćenje pomoćnih promenljivih

U situaciji kada je iskaznu formulu u DNF-u (formula oblika $\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} p_{ij}$, gde su p_{ij} literali) potrebno prebaciti u KNF, kako bi se smanjio rezultujući broj klauza moguće je iskoristiti pomoćne promenljive. Ovo dolazi uz cenu održavanja ekvizadovoljivosti[1] formule, ali ne i ekvivalentnosti. Konverzija se vrši na sledeći način:

$$\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m} p_{ij} \mapsto \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} (p_{ij} \vee \neg h_i)$$

pri čemu je potrebno dodati i disjunkciju pomoćnih promenljivih:

$$\bigvee_{i=1}^{n} h_i$$

Ekvivalencijski oblik ograničenja

Neka primitivna FlatZinc ograničenja se javljaju i u ekvivalencijskom obliku (engl. *reified*):

$$constr \Leftrightarrow r$$

gde je *constr* FlatZinc primitivno ograničenje, a r logička FlatZinc promenljiva.

Ako pretpostavimo da je ograničenje *constr* kodirano pomoću sledeće KNF formule:

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{m} p_{ij}$$

onda se njegov ekvivalencijski oblik kodira po sledećem principu (simbol $\mapsto$ označava kodiranje):

[1] Dve iskazne formule su ekvizadovoljive ukoliko su obe zadovoljive ili obe nezadovoljive

$$\bigwedge_{i=1}^{n}\bigvee_{j=1}^{m} p_{ij} \Leftrightarrow r \mapsto \tag{3.1}$$

$$(\neg r \wedge \bigvee_{i=1}^{n}\bigwedge_{j=1}^{m} \neg p_{ij}) \vee (r \wedge \bigwedge_{i=1}^{n}\bigvee_{j=1}^{m} p_{ij}) \tag{3.2}$$

Leva strana disjunkcije (3.2) doprinosi sledećim klauzama:

$$\neg r \vee \neg H_1$$
$$\bigwedge_{i=1}^{n}\bigwedge_{j=1}^{m} (\neg p_{ij} \vee \neg h_i)$$
$$(\bigvee_{i=1}^{n} h_i) \vee \neg H_1$$

Desna strana disjunkcije (3.2) doprinosi sledećim klauzama:

$$r \vee \neg H_2$$
$$\bigwedge_{i=1}^{n} (\bigvee_{j=1}^{m} p_{ij}) \vee \neg H_2$$

Najzad, kako mora važiti leva ili desna strana disjunkcije (3.2), dodaje se sledeća klauza (pri čemu su sa H_1 i H_2 označene pomoćne promenljive):

$$H_1 \vee H_2$$

Problem nastaje ukoliko se među klauzama ograničenja koje se prevodi u ekvivalencijski oblik nalaze pomoćne promenljive. Kako korišćenje pomoćnih promenljivih održava ekvizadovoljivost, ali ne i ekvivalentnost formule, gorenavedenim pristupom se neće očuvati ekvivalentnost između zadovoljenosti ograničenja i vrednosti promenljive r. Ovo je rešeno ili raspisivanjem originalnog ograničenja bez pomoćnih promenljivih, po cenu većeg broja klauza, ili prilagođavanjem pomoćnih promenljivih tako da održe ekvivalentnost formule (na primer, umesto $p_{ij} \vee \neg h_i$ koristimo $(p_{ij} \vee \neg h_i) \wedge (\neg p_{ij} \vee h_i)$).

Pored ekvivalencijskog, postoji i implikacijski oblik ograničenja:

$$r \Rightarrow constr$$

Kodira se relativno jednostavno, pošto se implikacija može lako svesti na disjunkciju:

$$r \Rightarrow \bigwedge_{i=1}^{n} \bigvee_{j=1}^{m} p_{ij} \mapsto \bigwedge_{i=1}^{n} (\bigvee_{j=1}^{m} p_{ij}) \vee \neg r$$

Vredi napomenuti da je i ovde potrebno voditi računa da li se u ograničenju koje se prevodi u implikacijski oblik koriste pomoćne promenljive.

Kodiranje supstitucija

U nekoliko ograničenja, deo postupka kodiranja uključuje kodiranje supstitucija oblika $x = c_1 \cdot x_1 + c_2 \cdot x_2$, pri čemu je x novouvedena promenljiva. Prvi korak je određivanje granica domena nove promenljive:

$$l(x) = min(c_1 \cdot l(x_1), c_1 \cdot u(x_1)) + min(c_2 \cdot l(x_2), c_2 \cdot u(x_2))$$
$$u(x) = max(c_1 \cdot l(x_1), c_1 \cdot u(x_1)) + max(c_2 \cdot l(x_2), c_2 \cdot u(x_2))$$

Nakon toga, supstitucija se kodira po sledećem principu, detaljnije objašnjenom u literaturi [12]:

$$-x + c_1 \cdot x_1 + c_2 \cdot x_2 \leq 0 \quad \wedge \quad x - c_1 \cdot x_1 - c_2 \cdot x_2 \leq 0$$

Prvi konjunkt kodira se na sledeći način:

$$\bigwedge_{i+j+k=-2} (\neg p_{x,-i-1} \vee p_{x_1,a} \vee p_{x_2,b})$$

pri čemu $i \in \{-u(x) - 1, \ldots, -l(x)\}$, $j \in \{l(c_1 \cdot x_1) - 1, \ldots, u(c_1 \cdot x_1)\}$ i $k \in \{l(c_2 \cdot x_2) - 1, \ldots, u(c_2 \cdot x_2)\}$, a parametri a i b se određuju na osnovu koeficijenata c_1 i c_2. Ukoliko važi $c_1 > 0$, onda je $a = \lfloor j/c_1 \rfloor$, a inače $a = \lceil j/c_1 \rceil - 1$ i literal koji uključuje a je negiran. Ukoliko važi $c_2 > 0$, onda je $b = \lfloor k/c_2 \rfloor$, a inače $b = \lceil k/c_2 \rceil - 1$ i literal koji uključuje b je negiran.

Drugi konjunkt kodira se na način sličan prvom:

$$\bigwedge_{i+j+k=-2} (p_{x,i} \vee p_{x_1,a} \vee p_{x_2,b})$$

pri čemu $i \in \{l(x) - 1, \ldots, u(x)\}$, $j \in \{l(-c_1 \cdot x_1) - 1, \ldots, u(-c_1 \cdot x_1)\}$ i $k \in \{l(-c_2 \cdot x_2) - 1, \ldots, u(-c_2 \cdot x_2)\}$, a parametri a i b se određuju na osnovu koeficijenata

c_1 i c_2. Ukoliko važi $c_1 < 0$, onda je $a = -\lfloor j/c_1 \rfloor$, a inače $a = -\lceil j/c_1 \rceil - 1$ i literal koji uključuje a je negiran. Ukoliko važi $c_2 < 0$, onda je $b = -\lfloor k/c_2 \rfloor$, a inače $b = -\lceil k/c_2 \rceil - 1$ i literal koji uključuje b je negiran.

Korišćenje parametara umesto promenljivih

U FlatZinc modelima postoji mogućnost da se na mestu argumenta koji je naznačen kao promenljiva nađe parametar, ali ne i obratno. Ovo je rešeno uvođenjem pomoćnih FlatZinc promenljivih koje ne učestvuju u ispisu i imaju fiksirane vrednosti.

Ukoliko celobrojni parametar ima vrednost a, kodira se na sledeći način:

$$p_{x,a}$$

$$\neg p_{x,a-1}$$

Ukoliko logički parametar ima vrednost $true$, biće kodiran jediničnom klauzom q_{pom}, odnosno ukoliko ima vrednost $false$ jediničnom klauzom $\neg q_{pom}$.

Ukoliko skupovni parametar ima vrednosti $a_1, \ldots, a_n$, biće kodiran jediničnim klauzama:

$$r_{pom,a_1}$$

$$\vdots$$

$$r_{pom,a_n}$$

3.3 Celobrojna ograničenja

Kao što je pomenuto u prethodnom odeljku, u ovom odeljku biće izloženi načini kodiranja pojedinačnih celobrojnih ograničenja.

Ograničenje int_le

Ovo ograničenje ima oblik:

$$int\_le(var\ int : a,\ var\ int : b)$$

i zahteva da važi nejednakost $a \leq b$. Može se svesti na kodiranje primitivne razlike $a - b \leq 0$.

Ograničenje int_lt

Ovo ograničenje ima oblik:

$$int\_lt(var\ int : a,\ var\ int : b)$$

i zahteva da važi nejednakost $a < b$. Može se svesti na kodiranje primitivne razlike $a - b \leq -1$.

Ograničenje int_eq

Ovo ograničenje ima oblik:

$$int\_eq(var\ int : a,\ var\ int : b)$$

i zahteva da važi jednakost $a = b$. Može se svesti na kodiranje ograničenja $int\_le(a, b)$ i $int\_le(b, a)$, a zatim konjunkciju dobijenih KNF formula.

Ograničenje int_ne

Ovo ograničenje ima oblik:

$$int\_ne(var\ int : a,\ var\ int : b)$$

i zahteva da važi nejednakost $a \neq b$. Može se svesti na kodiranje ograničenja $int\_lt(a, b)$ i $int\_lt(b, a)$, a zatim disjunkciju dobijenih KNF formula. Kako bi dobijena formula ostala u KNF-u, mogu se iskoristiti dve pomoćne promenljive h_1 i h_2. KNF formula dobijena kodiranjem prvog ograničenja prevodi se u njoj ekvizadovoljivu formulu:

$$\bigwedge_{i=1}^{n_1} \bigvee_{j=1}^{m_1} p_{ij} \mapsto \bigwedge_{i=1}^{n_1} (\neg h_1 \vee \bigvee_{j=1}^{m_1} p_{ij})$$

Slično važi i za KNF formulu dobijenu kodiranjem drugog ograničenja:

$$\bigwedge_{i=1}^{n_2} \bigvee_{j=1}^{m_2} p_{ij} \mapsto \bigwedge_{i=1}^{n_2} (\neg h_2 \vee \bigvee_{j=1}^{m_2} p_{ij})$$

Najzad, kako mora biti zadovoljeno barem jedno od dva ograničenja, dodaje se sledeća klauza:

$$h_1 \vee h_2$$

Ograničenje int_max

Ovo ograničenje ima oblik:

$$int\_max(var\ int : a,\ var\ int : b,\ var\ int : c)$$

i zahteva da važi $max(a, b) = c$. Ideja kodiranja je sledeća:

$$a \leq c \quad \wedge \quad b \leq c \quad \wedge \quad (c \leq a \vee c \leq b)$$

Može se svesti na konjunkciju KNF formula dobijenih kodiranjem ograničenja $int\_le(a, c)$ i $int\_le(b, c)$, a zatim konjunkciju te formule sa disjunkcijom KNF formula dobijenih kodiranjem ograničenja $int\_le(c, a)$ i $int\_le(c, b)$. Kako bi pomenuta disjunkcija ostala u KNF-u, moguće je upotrebiti pomoćne promenljive, slično kao kod ograničenja int_ne.

Ograničenje int_min

Ovo ograničenje ima oblik:

$$int\_min(var\ int : a,\ var\ int : b,\ var\ int : c)$$

i zahteva da važi $min(a, b) = c$. Ideja kodiranja je sledeća:

$$c \leq a \quad \wedge \quad c \leq b \quad \wedge \quad (a \leq c \vee b \leq c)$$

Može se svesti na konjunkciju KNF formula dobijenih kodiranjem ograničenja $int\_le(c, a)$ i $int\_le(c, b)$, a zatim konjunkciju te formule sa disjunkcijom KNF formula dobijenih kodiranjem ograničenja $int\_le(a, c)$ i $int\_le(b, c)$. Kako bi pomenuta disjunkcija ostala u KNF-u, moguće je upotrebiti pomoćne promenljive, slično kao kod ograničenja int_ne.

Ograničenje int_abs

Ovo ograničenje ima oblik:

$$int\_abs(var\ int : a,\ var\ int : b)$$

i zahteva da važi $|a| = b$. Ideja kodiranja je sledeća:

$$max(a, -a) = b$$

što je ekvivalentno sa:

$$a \leq b \quad \wedge \quad -a \leq b \quad \wedge \quad (b \leq a \vee b \leq -a)$$

Prvi konjunkt se može dobiti kodiranjem ograničenja $int\_le(a, b)$. Drugi konjunkt se svodi na sledeću nejednakost: $a + b \geq 0$; odnosno na negaciju kodiranja primitivnog zbira $a+b \leq -1$. Kako se negacijom KNF formule dobija DNF formula, koriste se pomoćne promenljive da bi se održao KNF oblik. Treći konjunkt je zapravo disjunkcija formula dobijenih kodiranjem ograničenja $int\_le(a, b)$ i primitivnog zbira $a+b \leq 0$. Kako bi pomenuta disjunkcija ostala u KNF-u, moguće je upotrebiti pomoćne promenljive, slično kao kod ograničenja int_ne.

Ograničenje int_plus

Ovo ograničenje ima oblik:

$$int\_plus(var\ int : a,\ var\ int : b,\ var\ int : c)$$

i zahteva da važi jednakost $a+b = c$. Svodi se na kodiranje supstitucije $c = c_1 \cdot a + c_2 \cdot b$, pri čemu su oba koeficijenta jednaka 1, a granice domena promenljive c su unapred poznate.

Ograničenje array_int_element

Ovo ograničenje ima oblik:

$$array\_int\_element(var\ int : b,\ array\ [int]\ of\ int : as,\ var\ int : c)$$

i zahteva da važi jednakost $as[b] = c$, pri čemu je as niz celobrojnih konstanti. Ideja kodiranja je sledeća:

$$\bigvee_{i \in D(b) \cap I(as)} b = i \wedge c = as[i]$$

pri čemu je $I(as)$ skup indeksa niza as. Jednakost $b = i$ kodira se pomoću dve jedinične klauze:

$$p_{b,i}$$

$$\neg p_{b,i-1}$$

Analogno tome se kodira i jednakost $c = as[i]$. Kako bi formula bila u KNF obliku, koriste se pomoćne promenljive.

Ograničenje array_int_var_element

Ovo ograničenje ima oblik:

$$array\_int\_var\_element(var\ int : b,\ array\ [int]\ of\ var\ int : as,\ var\ int : c)$$

i zahteva da važi jednakost $as[b] = c$, pri čemu je as niz celobrojnih promenljivih. Kodira se analogno ograničenju $array\_int\_element$, osim što se jednakost $c = as[i]$ ovog puta svodi na kodiranje ograničenja $int\_eq(c, as[i])$.

Ograničenje array_int_maximum

Ovo ograničenje ima oblik:

$$array\_int\_maximum(var\ int : m,\ array\ [int]\ of\ int : x)$$

i zahteva da promenljiva m bude jednaka maksimalnoj vrednosti iz niza x. Ideja kodiranja je sledeća:

$$\bigwedge_{i \in I(x)} m \geq x[i] \quad \wedge \quad (\bigvee_{i \in I(x)} m \leq x[i]))$$

Poređenja $m \geq x[i]$ i $m \leq x[i]$ svode se na kodiranje ograničenja $int\_le(x[i], m)$, odnosno $int\_le(m, x[i])$. Kako bi disjunkcija sa desne strane ostala u KNF-u, koriste se pomoćne promenljive.

Ograničenje array_int_minimum

Ovo ograničenje ima oblik:

$$array\_int\_minimum(var\ int : m,\ array\ [int]\ of\ int : x)$$

i zahteva da promenljiva m bude jednaka minimalnoj vrednosti iz niza x. Ideja kodiranja je sledeća:

$$\bigwedge_{i \in I(x)} m \leq x[i] \quad \wedge \quad (\bigvee_{i \in I(x)} m \geq x[i]))$$

Poređenja $m \leq x[i]$ i $m \geq x[i]$ svode se na kodiranje ograničenja $int\_le(m, x[i])$, odnosno $int\_le(x[i], m)$. Kako bi disjunkcija sa desne strane ostala u KNF-u, koriste se pomoćne promenljive.

Ograničenje int_times

Ovo ograničenje ima oblik:

$$int\_times(var\ int : a,\ var\ int : b,\ var\ int : c)$$

i zahteva da važi jednakost $a \cdot b = c$. U osnovi kodiranja nalazi se ideja da se vrednost promenljive c ograniči vrednostima promenljivih a i b. U slučaju da su svi elementi domena činilaca nenegativni, usled monotonosti operacije množenja, ovo je moguće postići ispunjavanjem sledećih uslova:

$$\forall i \in D(a), j \in D(b) \quad (a \leq i \wedge b \leq j \Rightarrow c \leq i \cdot j) \tag{3.3}$$

$$\forall i \in D(a), j \in D(b) \quad (a \geq i \wedge b \geq j \Rightarrow c \geq i \cdot j) \tag{3.4}$$

Gorenavedeni uslovi ekvivalentni su sledećim klauzama:

$$\forall i \in D(a), j \in D(b) \quad (\neg p_{a,i} \vee \neg p_{b,j} \vee p_{c,i \cdot j}) \tag{3.5}$$

$$\forall i \in D(a), j \in D(b) \quad (p_{a,i-1} \vee p_{b,j-1} \vee \neg p_{c,i \cdot j-1}) \tag{3.6}$$

U slučaju da je $i \cdot j \geq u(c)$ formula (3.3) je tautologija, pa se ne mora dodavati. Slično važi za formulu (3.4) kada je ispunjen uslov $i \cdot j \leq l(c)$.

U slučaju da je $i \cdot j < l(c)$, promenljiva $p_{c,i \cdot j}$ će biti uvek netačna, pa se ne mora dodavati u formulu (3.3). Slično važi za formulu (3.4) kada je ispunjen uslov $i \cdot j > u(c)$; literal $\neg p_{c,i \cdot j-1}$ će biti uvek netačan, pa se ne mora dodavati u formulu.

Ukoliko nisu svi elementi domena činilaca isključivo nenegativni, uslove (3.1) i (3.2) je potrebno preformulisati:

$$(\forall m_i, M_i \in D(a))(\forall m_j, M_j \in D(b)) \quad (m_i \leq a \leq M_i \wedge m_j \leq b \leq M_j \Rightarrow c \leq M_c)$$

$$(\forall m_i, M_i \in D(a))(\forall m_j, M_j \in D(b)) \quad (m_i \leq a \leq M_i \wedge m_j \leq b \leq M_j \Rightarrow c \geq m_c)$$

gde je $M_c = max(m_i \cdot m_j, m_i \cdot M_j, M_i \cdot m_j, M_i \cdot M_j)$, $m_c = min(m_i \cdot m_j, m_i \cdot M_j, M_i \cdot m_j, M_i \cdot M_j)$ i važi $m_i \leq M_i$, odnosno $m_j \leq M_j$.

Dobijene klauze su ovaj put sledeće:

$$p_{a,m_i-1} \vee \neg p_{a,M_i} \vee p_{b,m_j-1} \vee \neg p_{b,M_j} \vee p_{c,M_c} \tag{3.7}$$

$$p_{a,m_i-1} \vee \neg p_{a,M_i} \vee p_{b,m_j-1} \vee \neg p_{b,M_j} \vee \neg p_{c,m_c-1} \tag{3.8}$$

Formula (3.5) se može obrisati u slučaju da je $u(c) \leq M_c$, a u slučaju da važi $l(c) > M_c$, literal p_{c,M_c} se može ukloniti iz nje.

Slično, formula (3.6) se može obrisati u slučaju da je $l(c) \geq m_c$, a u slučaju da važi $u(c) < M_c$, literal p_{c,m_c-1} se može ukloniti iz nje.

Ograničenje int_div

Ovo ograničenje ima oblik:

$$int\_div(var\ int : a,\ var\ int : b,\ var\ int : c)$$

i zahteva da važi jednakost $a/b = c$, gde je / operator celobrojnog deljenja. Iz jednakosti $a/b = c$ sledi:

$$a = b \cdot c + r$$

za neko $r \in \{0, \ldots, b-1\}$. Dakle, u slučaju pozitivnih domena promenljivih, bilo bi dovoljno kodirati sledeće uslove:

$$a = b \cdot c + r \quad \wedge \quad r \geq 0 \quad \wedge \quad r \leq b - 1$$

Ipak, kako domeni celobrojnih promenljivih mogu uključivati negativne vrednosti, treba voditi računa na koji način „zaokružiti" količnik. Npr. $-5/2$ može se tumačiti kao -2 ili -3 u zavisnosti od znaka ostatka. Kako je u jeziku MiniZinc generalno praksa da ostatak ima isti znak kao deljenik, u gorenavedenom slučaju ostatak bi bio -1, a količnik -2. Kako bi se ispoštovao ovaj uslov, kodiranje ide po sledećem principu:

$$
\begin{aligned}
& a = b \cdot c + r \quad \wedge \\
& |r| < |b| \quad \wedge \\
& a \geq 0 \Rightarrow r \geq 0 \quad \wedge \\
& a \leq 0 \Rightarrow r \leq 0
\end{aligned}
$$

Prvi konjunkt se dobija uvođenjem pomoćne celobrojne promenljive bc i ograničenja $int\_times(b, c, bc)$, a zatim i ograničenja $int\_plus(bc, r, a)$. Drugi konjunkt se dobija uvođenjem pomoćnih celobrojnih promenljivih r' i b', a zatim i ograničenja $int\_abs(r, r')$, $int\_abs(b, b')$ i $int\_lt(r', b')$. Treći i četvrti konjunkt svode se na jednostavne disjunkcije literala:

$$
\begin{aligned}
& p_{a,-1} \vee \neg p_{r,-1} \\
& \neg p_{a,0} \vee p_{r,0}
\end{aligned}
$$

Ograničenje int_mod

Ovo ograničenje ima oblik:

$$int\_mod(var\ int : a,\ var\ int : b,\ var\ int : c)$$

i zahteva da važi jednakost $a\%b = c$, gde je $\%$ operator računanja ostatka pri celobrojnom deljenju. Kodira se analogno ograničenju int_div, osim što ulogu ostatka r preuzima promenljiva c, a količnik je ovog puta pomoćna promenljiva p.

Ograničenje int_pow

Ovo ograničenje ima oblik:

$$int\_pow(var\ int : x,\ var\ int : y,\ var\ int : z)$$

i zahteva da važi jednakost $x^y = z$. Za kodiranje ovog ograničenja korišćeno je retko kodiranje:

$$\bigvee_{i \in D(x), j \in D(y)} s_{x,i} \wedge s_{y,j} \wedge s_{z,i^j}$$

Kako je navedena formula u DNF-u, korišćene su pomoćne promenljive kako bi bila prebačena u KNF. Takođe, nema potrebe dodavati klauze kod kojih $i^j \notin D(z)$. Specijalno, u slučaju da ne postoje $i \in D(x)$ i $j \in D(y)$ za koje važi $i^j \in D(z)$, gornja disjunkcija postaje prazna. Na taj način nastaje prazna klauza, što je ekvivalentno sa $\bot$, pa formula postaje nezadovoljiva.

Ograničenje int_lin_le

Ovo ograničenje ima oblik:

$$int\_lin\_le(array\ [int]\ of\ int : as,\ array\ [int]\ of\ var\ int : bs,\ int : c)$$

i zahteva da važi nejednakost $\sum_{i=1}^{n} as[i] \cdot bs[i] \leq c$, pri čemu je as niz celobrojnih konstanti, bs niz celobrojnih promenljivih, a n broj elemenata u svakom od ova dva niza. Kodiranje ovog ograničenja oslanja se na supstitucije ([12]):

$$\begin{aligned} & x_1 = as[1] \cdot bs[1] + as[2] \cdot bs[2] \\ \forall i \in \{2 \ldots n-1\} \quad & x_i = x_{i-1} + as[i+1] \cdot bs[i+1] \end{aligned}$$

Na ovaj način postiže se da promenljiva x_{n-1} predstavlja sumu $\sum_{i=1}^{n} as[i] \cdot bs[i]$. Kako bi se ograničenje kodiralo, preostaje još kodiranje same nejednakosti, korišćenjem $int\_le(x_{n-1}, c)$.

Ograničenje int_lin_eq

Ovo ograničenje ima oblik:

$$int\_lin\_eq(array\ [int]\ of\ int : as,\ array\ [int]\ of\ var\ int : bs,\ int : c)$$

i zahteva da važi nejednakost $\sum_{i=1}^{n} as[i] \cdot bs[i] = c$, pri čemu je as niz celobrojnih konstanti, bs niz celobrojnih promenljivih, a n broj elemenata u svakom od ova dva niza. Kodira se analogno ograničenju int_lin_le, osim što se na kraju koristi ograničenje $int\_eq(x_{n-1}, c)$.

Ograničenje int_lin_ne

Ovo ograničenje ima oblik:

$$int\_lin\_ne(array\ [int]\ of\ int : as,\ array\ [int]\ of\ var\ int : bs,\ int : c)$$

i zahteva da važi nejednakost $\sum_{i=1}^{n} as[i] \cdot bs[i] \neq c$, pri čemu je as niz celobrojnih konstanti, bs niz celobrojnih promenljivih, a n broj elemenata u svakom od ova dva niza. Kodira se analogno ograničenju int_lin_le, osim što se na kraju koristi ograničenje $int\_ne(x_{n-1}, c)$.

Ekvivalencijski oblici ograničenja

Ograničenja: int_eq_reif, int_le_reif, int_lin_eq_reif, int_lin_le_reif, int_lin_ne_reif, int_lt_reif, int_ne_reif, kao i njihove verzije sa implikacijom (sufiks $\_imp$, umesto $\_reif$), kodiraju se ranije opisanim postupkom kodiranja ekvivalencijskih, odnosno implikacijskih oblika ograničenja.

3.4 Logička ograničenja

U ovom odeljku biće izloženi načini kodiranja pojedinačnih logičkih ograničenja.

Ograničenje bool_le

Ovo ograničenje ima oblik:

$$bool\_le(var\ bool : x,\ var\ bool : y)$$

i zahteva da važi $x \leq y$, pri čemu se smatra da važi uređenje u kome je vrednost logičke promenljive *tačno* veća od vrednosti *netačno*. Kodira se sledećom klauzom:

$$\neg q_x \vee q_y$$

Ograničenje bool_lt

Ovo ograničenje ima oblik:

$$bool\_lt(var\ bool : x,\ var\ bool : y)$$

i zahteva da važi $x < y$, pri čemu se smatra da važi isto uređenje kao u ograničenju $bool\_le$. Kodira se sledećim jediničnim klauzama:

$$\neg q_x$$

$$q_y$$

Ograničenje bool_eq

Ovo ograničenje ima oblik:

$$bool\_eq(var\ bool : x,\ var\ bool : y)$$

i zahteva da važi $x = y$. Kodira se na sledeći način:

$$(q_x \vee \neg q_y) \wedge (\neg q_x \vee q_y)$$

Ograničenje bool_not

Ovo ograničenje ima oblik:

$$bool\_not(var\ bool : x,\ var\ bool : y)$$

i zahteva da važi $x \neq y$. Kodira se na sledeći način:

$$(q_x \vee q_y) \wedge (\neg q_x \vee \neg q_y)$$

Ograničenje bool_xor

Ovo ograničenje ima dva različita oblika. Prvi oblik je:

$$bool\_xor(var\ bool : x,\ var\ bool : y)$$

i zahteva da izraz $x \oplus y$ bude tačan, pri čemu je $\oplus$ operator ekskluzivne disjunkcije. Kako je $x \oplus y$ tačno akko važi $x \neq y$, kodiranje ovog ograničenja svodi se na kodiranje ograničenja $bool\_not(x, y)$.

Drugi oblik ovog ograničenja je:

$$bool\_xor(var\ bool : a,\ var\ bool : b,\ var\ bool : r)$$

i zahteva da važi $r \Leftrightarrow a \oplus b$. Kodira se na sledeći način:

$$\begin{array}{lll} (q_a \vee q_b \vee \neg q_r) & \wedge & (\neg q_a \vee \neg q_b \vee \neg q_r) \quad \wedge \\ (\neg q_a \vee q_b \vee q_r) & \wedge & (q_a \vee \neg q_b \vee q_r) \end{array}$$

Ograničenje bool_and

Ovo ograničenje ima oblik:

$$bool\_and(var\ bool : a,\ var\ bool : b,\ var\ bool : r)$$

i zahteva da važi $r \Leftrightarrow a \wedge b$. Kodira se na sledeći način:

$$(q_a \vee \neg q_r) \wedge (q_b \vee \neg q_r) \wedge (\neg q_a \vee \neg q_b \vee q_r)$$

Ograničenje bool_or

Ovo ograničenje ima oblik:

$$bool\_or(var\ bool : a,\ var\ bool : b,\ var\ bool : r)$$

i zahteva da važi $r \Leftrightarrow a \vee b$. Kodira se na sledeći način:

$$(\neg q_a \vee q_r) \wedge (\neg q_b \vee q_r) \wedge (q_a \vee q_b \vee \neg q_r)$$

Ograničenje bool_clause

Ovo ograničenje ima oblik:

$$bool\_clause(array\ [int]\ of\ var\ bool : as,\ array\ [int]\ of\ var\ bool : bs)$$

i zahteva da disjunkcija $(\bigvee_{i=1}^{n} a[i]) \vee (\bigvee_{j=1}^{m} \neg b[j])$ bude tačna. Kako oba niza sadrže logičke promenljive, kodiranje se svodi na dodavanje jedne klauze:

$$(\bigvee_{i=1}^{n} q_{a[i]}) \vee (\bigvee_{j=1}^{m} \neg q_{b[j]})$$

Ograničenje array_bool_and

Ovo ograničenje ima oblik:

$$array\_bool\_and(array\ [int]\ of\ var\ bool : as,\ var\ bool : r)$$

i zahteva da važi $r \Leftrightarrow \bigwedge_{i=1}^{n} as[i]$. Navedeni izraz može se svesti na KNF oblik sledećim transformacijama:

$$r \Leftrightarrow \bigwedge_{i=1}^{n} as[i] \equiv$$
$$(\neg r \vee \bigwedge_{i=1}^{n} as[i]) \wedge (r \vee \neg \bigwedge_{i=1}^{n} as[i]) \equiv$$
$$\bigwedge_{i=1}^{n} (\neg r \vee as[i]) \wedge (r \vee \bigvee_{i=1}^{n} \neg as[i])$$

Za kodiranje krajnjeg izraza koriste se sledeće klauze:

$$\forall i \in \{1, \ldots, n\} \quad (\neg q_r \vee q_{as[i]})$$
$$q_r \vee \bigvee_{i=1}^{n} \neg q_{as[i]}$$

Ograničenje array_bool_or

Ovo ograničenje ima oblik:

$$array\_bool\_or(array\ [int]\ of\ var\ bool : as,\ var\ bool : r)$$

i zahteva da važi $r \Leftrightarrow \bigvee_{i=1}^{n} as[i]$. Navedeni izraz može se svesti na KNF oblik sledećim transformacijama:

$$r \Leftrightarrow \bigvee_{i=1}^{n} as[i] \equiv$$
$$(\neg r \vee \bigvee_{i=1}^{n} as[i]) \wedge (r \vee \neg \bigvee_{i=1}^{n} as[i]) \equiv$$
$$(\neg r \vee \bigvee_{i=1}^{n} as[i]) \wedge (r \vee \bigwedge_{i=1}^{n} \neg as[i]) \equiv$$
$$(\neg r \vee \bigvee_{i=1}^{n} as[i]) \wedge \bigwedge_{i=1}^{n} (r \vee \neg as[i])$$

Za kodiranje krajnjeg izraza koriste se sledeće klauze:

$$\forall i \in \{1, \ldots, n\} \quad (q_r \vee \neg q_{as[i]})$$
$$\neg q_r \vee \bigvee_{i=1}^{n} q_{as[i]}$$

Ograničenje array_bool_element

Ovo ograničenje ima oblik:

$$array\_bool\_element(var\ int : b,\ array\ [int]\ of\ bool : as,\ var\ bool : c)$$

i zahteva da važi $as[b] = c$, pri čemu je as niz logičkih konstanti. Ideja kodiranja je sledeća:

$$\bigvee_{i \in D(b) \cap I(as)} b = i \wedge c = as[i]$$

pri čemu je $I(as)$ skup indeksa niza as. Jednakost $b = i$ kodira se pomoću dve jedinične klauze:

$$p_{b,i}$$

$$\neg p_{b,i-1}$$

Kodiranje jednakosti $c = as[i]$ svodi se na proveru vrednosti konstante $as[i]$. Ukoliko je ta vrednost *tačno*, dodaje se jedinična klauza q_c. U suprotnom, dodaje se jedinična klauza $\neg q_c$. Kako bi formula bila u KNF obliku, koriste se pomoćne promenljive.

Ograničenje array_var_bool_element

Ovo ograničenje ima oblik:

$$array\_var\_bool\_element(var\ int : b,\ array\ [int]\ of\ var\ bool : as,\ var\ bool : c)$$

i zahteva da važi $as[b] = c$, pri čemu je as niz logičkih promenljivih. Kodira se analogno ograničenju $array\_bool\_element$, osim što se jednakost $c = as[i]$ ovog puta svodi na kodiranje ograničenja $bool\_eq(c, as[i])$.

Ograničenje bool_lin_eq

Ovo ograničenje ima oblik:

$$bool\_lin\_eq(array\ [int]\ of\ int : as,\ array\ [int]\ of\ var\ bool : bs,\ var\ int : c)$$

i zahteva da važi jednakost $\sum_{i=1}^{n} as[i] \cdot bs[i] = c$, pri čemu je as niz celobrojnih konstanti, bs niz logičkih promenljivih, a n broj elemenata u svakom od ova dva

niza. U praksi, niz as najčešće čine vrednosti 0 i 1, pa se ovo ograničenje svodi na ograničenje kardinalnosti - vrednost promenljive c diktira koliko će elemenata niza bs uz koje je koeficijent 1 biti tačno. Kodiranje ovog ograničenja oslanja se na supstitucije, slično celobrojnim linearnim ograničenjima:

$$\begin{aligned} x_1 &= as[1] \cdot bs[1] + as[2] \cdot bs[2] \\ \forall i \in \{2 \dots n-1\} \quad x_i &= x_{i-1} + as[i+1] \cdot bs[i+1] \end{aligned}$$

Pri čemu se u ovom kontekstu logičke promenljive $bs[i]$ posmatraju kao celobrojne promenljive sa domenom $\{0, 1\}$. Takođe, pri kodiranju pojedinačnih supstitucija, za pomoćne promenljive x_i korišćeno je retko kodiranje, pa kodiranje supstitucije $x = c_1 \cdot x_1 + c_2 \cdot x_2$ funkcioniše po sledećem principu:

$$\bigvee_{i=c_1 \cdot j + c_2 \cdot k} (s_{x,i} \wedge s_{x_1,j} \wedge s_{x_2,k})$$

pri čemu važi $i \in D(x)$, $j \in D(x_1)$ i $k \in D(x_2)$. Gorenavedena formula je u DNF-u, pa se za konverziju u KNF koriste pomoćne promenljive. Treba napomenuti da se, u slučaju da je x_1 ili x_2 zapravo jedna od logičkih promenljivih $bs[i]$, umesto iskaznih promenljivih $s_{x_1,j}$(ili $s_{x_2,k}$) koriste literali $q_{bs[i]}$ ili $\neg q_{bs[i]}$, u zavisnosti od toga da li je j (ili k) jednako 1 ili 0, respektivno.

Kako bi se ograničenje kodiralo, preostaje još kodiranje same jednakosti, korišćenjem ograničenja $int\_eq(x_{n-1}, c)$.

Ograničenje bool_lin_le

Ovo ograničenje ima oblik:

$$bool\_lin\_le(array\ [int]\ of\ int : as,\ array\ [int]\ of\ var\ bool : bs,\ var\ int : c)$$

i zahteva da važi nejednakost $\sum_{i=1}^{n} as[i] \cdot bs[i] \neq c$, pri čemu je as niz celobrojnih konstanti, bs niz logičkih promenljivih, a n broj elemenata u svakom od ova dva niza. Kodira se analogno ograničenju $bool\_lin\_eq$, osim što se na kraju koristi ograničenje $int\_le(x_{n-1}, c)$.

Ograničenje array_bool_xor

Ovo ograničenje ima oblik:

$$array\_bool\_xor(array\ [int]\ of\ var\ bool : as)$$

i zahteva da formula $\oplus_{i=1}^{n} as[i]$ bude tačna, pri čemu je n broj elemenata niza as. Pri kodiranju, uvode se sledeće pomoćne promenljive:

$$\begin{aligned} & r_1 = as[1] \oplus as[2] \\ \forall i \in \{2 \dots n-2\} \quad & r_i = r_{i-1} \oplus as[i+1] \end{aligned}$$

pri čemu se kodiranje jednakosti $r_1 = as[1] \oplus as[2]$ svodi na kodiranje ograničenja $bool\_xor(as[1], as[2], r_1)$. Po završetku kodiranja gorenavedenih jednakosti, u promenljivoj r_{n-2} sačuvana je vrednost izraza $\oplus_{i=1}^{n-1} as[i]$. Kako bi se osiguralo da formula $\oplus_{i=1}^{n} as[i]$ bude tačna, potrebno je obezbediti da vrednosti promenljivih r_{n-2} i $as[n]$ budu različite. Ovo se može postići kodiranjem ograničenja $int\_xor(r_{n-2}, as[n])$.

Ograničenje bool2int

Ovo ograničenje ima oblik:

$$bool2int(var\ bool : a,\ var\ int : b)$$

i zahteva da se domen promenljive b svede na $\{0, 1\}$ i da važi ekvivalencija $a \Leftrightarrow (b = 1)$. Može se kodirati na sledeći način:

$$(\neg a \wedge (b = 0)) \quad \vee \quad (a \wedge (b = 1))$$

što je ekvivalentno sa:

$$(\neg q_a \wedge p_{b,0} \wedge \neg p_{b,-1}) \quad \vee \quad (q_a \wedge p_{b,1} \wedge \neg p_{b,0})$$

Gorenavedena formula je u DNF-u, pa se korišćenjem dve pomoćne promenljive svodi na KNF.

Ograničenje bool_eq_reif

Ovo ograničenje ima oblik:

$$bool\_eq\_reif(var\ bool : a,\ var\ int : b,\ var\ int : r)$$

i zahteva da važi $r \Leftrightarrow (a = b)$. Kodira se dodavanjem sledećih klauza:

$$q_r \vee \neg q_a \vee \neg q_b$$
$$q_r \vee q_a \vee q_b$$
$$\neg q_r \vee q_a \vee \neg q_b$$
$$\neg q_r \vee \neg q_a \vee q_b$$

Ograničenje bool_le_reif

Ovo ograničenje ima oblik:

$$bool\_le\_reif(var\ bool : a,\ var\ int : b,\ var\ int : r)$$

i zahteva da važi $r \Leftrightarrow (a \leq b)$. Kodira se dodavanjem sledećih klauza:

$$\neg q_r \vee \neg q_a \vee q_b$$
$$q_r \vee q_a$$
$$q_r \vee \neg q_b$$

Ograničenje bool_lt_reif

Ovo ograničenje ima oblik:

$$bool\_lt\_reif(var\ bool : a,\ var\ int : b,\ var\ int : r)$$

i zahteva da važi $r \Leftrightarrow (a < b)$. Kodira se dodavanjem sledećih klauza:

$$q_r \vee q_a \vee \neg q_b$$
$$\neg q_r \vee \neg q_a$$
$$\neg q_r \vee q_b$$

3.5 Skupovna ograničenja

U ovom odeljku biće izloženi načini kodiranja pojedinačnih skupovnih ograničenja.

Ograničenje set_in

Ovo ograničenje ima dva različita oblika. Prvi oblik je:

$$set\_in(var\ int : x,\ set\ of\ int : S)$$

i zahteva da važi $x \in S$, pri čemu je S skupovni parametar. Ideja kodiranja je sledeća:

$$\bigvee_{i \in S \cap D(x)} x = i$$

što je ekvivalentno sa:

$$\bigvee_{i \in S \cap D(x)} p_{x,i} \wedge \neg p_{x,i-1}$$

Kako je formula iznad u DNF-u, za prevođenje u KNF se koriste pomoćne promenljive.

Drugi oblik ovog ograničenja je:

$$set\_in(var\ int : x,\ var\ set\ of\ int : S)$$

i zahteva da važi $x \in S$, pri čemu je S skupovna promenljiva. Ideja kodiranja je sledeća:

$$\bigvee_{i \in A_S \cap D(x)} (x = i) \wedge (i \in S) \tag{3.9}$$

Podsetimo se, domen skupovne promenljive S se u jeziku FlatZinc definiše kao partitivni skup nekog datog skupa A_S, tj. $D(S) = P(A_S)$. Formula (3.9) je ekvivalentna sa:

$$\bigvee_{i \in A_S \cap D(x)} p_{x,i} \wedge \neg p_{x,i-1} \wedge r_{S,i}$$

Kako je formula iznad u DNF-u, za prevođenje u KNF se koriste pomoćne promenljive.

Ograničenje set_eq

Ovo ograničenje ima oblik:

$$set\_eq(var\ set\ of\ int : x,\ var\ set\ of\ int : y)$$

i zahteva da važi $x = y$. Kodiranje se vrši na sledeći način:

$$\begin{aligned}(\forall i \in A_x \setminus A_y) \quad & \neg r_{x,i} \\ (\forall i \in A_y \setminus A_x) \quad & \neg r_{y,i} \\ (\forall i \in A_x \cap A_y) \quad & (r_{x,i} \vee \neg r_{y.i}) \wedge (\neg r_{x,i} \vee r_{y,i})\end{aligned}$$

Ograničenje set_ne

Ovo ograničenje ima oblik:

$$set\_ne(var\ set\ of\ int : x,\ var\ set\ of\ int : y)$$

i zahteva da važi $x \neq y$. Kodiranje se vrši na sledeći način:

$$\begin{gathered}(\forall i \in A_x \cap A_y) \quad (r_{x,i} \vee r_{y.i} \vee \neg h_i) \wedge (\neg r_{x,i} \vee \neg r_{y,i} \vee \neg h_i) \\ (\bigvee_{i \in A_x \setminus A_y} r_{x,i}) \vee (\bigvee_{j \in A_y \setminus A_x} r_{y,j}) \vee (\bigvee_{k \in A_x \cup A_y} h_k)\end{gathered}$$

Ograničenje set_subset

Ovo ograničenje ima oblik:

$$set\_subset(var\ set\ of\ int : x,\ var\ set\ of\ int : y)$$

i zahteva da važi $x \subseteq y$. Kodiranje se vrši na sledeći način:

$$\begin{aligned}(\forall i \in A_x \setminus A_y) \quad & \neg r_{x,i} \\ (\forall i \in A_x \cap A_y) \quad & (\neg r_{x,i} \vee r_{y,i})\end{aligned}$$

Ograničenje set_superset

Ovo ograničenje ima oblik:

$$set\_superset(var\ set\ of\ int : x,\ var\ set\ of\ int : y)$$

i zahteva da važi $x \supseteq y$. Svodi se na kodiranje ograničenja $set\_subset(y, x)$.

Ograničenje set_intersect

Ovo ograničenje ima oblik:

$$set\_intersect(var\ set\ of\ int : x,\ var\ set\ of\ int : y,\ var\ set\ of\ int : z)$$

i zahteva da važi $z = x \cap y$. Kodira se na sledeći način:

$$\begin{aligned}
(\forall i \in A_x \setminus A_y) \quad & \neg r_{z,i} \\
(\forall i \in A_y \setminus A_x) \quad & \neg r_{z,i} \\
(\forall i \in A_x \cap A_y) \quad & r_{z,i} \Leftrightarrow (r_{x,i} \wedge r_{y,i})
\end{aligned}$$

pri čemu se kodiranje gornje ekvivalencije svodi na kodiranje ograničenja $bool\_and(r_{x,i}, r_{y,i}, r_{z,i})$.

Ograničenje set_union

Ovo ograničenje ima oblik:

$$set\_union(var\ set\ of\ int : x,\ var\ set\ of\ int : y,\ var\ set\ of\ int : z)$$

i zahteva da važi $z = x \cup y$. Kodira se na sledeći način:

$$\begin{aligned}
(\forall i \in A_x \setminus A_y) \quad & (r_{x,i} \vee \neg r_{z.i}) \wedge (\neg r_{x,i} \vee r_{z,i}) \\
(\forall i \in A_y \setminus A_x) \quad & (r_{y,i} \vee \neg r_{z.i}) \wedge (\neg r_{y,i} \vee r_{z,i}) \\
(\forall i \in A_x \cap A_y) \quad & r_{z,i} \Leftrightarrow (r_{x,i} \vee r_{y,i})
\end{aligned}$$

pri čemu se kodiranje gornje ekvivalencije svodi na kodiranje ograničenja $bool\_or(r_{x,i}, r_{y,i}, r_{z,i})$.

Ograničenje set_diff

Ovo ograničenje ima oblik:

$$set\_diff(var\ set\ of\ int : x,\ var\ set\ of\ int : y,\ var\ set\ of\ int : z)$$

i zahteva da važi $z = x \setminus y$. Kodira se na sledeći način:

$$\begin{aligned}
(\forall i \in A_x \setminus A_y) \quad & (r_{x,i} \vee \neg r_{z.i}) \wedge (\neg r_{x,i} \vee r_{z,i}) \\
(\forall i \in A_x \cap A_y) \quad & (\neg r_{z,i} \vee r_{x,i}) \wedge (\neg r_{z,i} \vee \neg r_{y,i}) \wedge (r_{z,i} \vee \neg r_{x,i} \vee r_{y,i})
\end{aligned}$$

Ograničenje set_symdiff

Ovo ograničenje ima oblik:

$$set\_symdiff(var\ set\ of\ int : x,\ var\ set\ of\ int : y,\ var\ set\ of\ int : z)$$

i zahteva da z bude simetrična razlika skupova x i y, odnosno da važi $z = (x \setminus y) \cup (y \setminus x)$. Kodira se na sledeći način:

$$\begin{aligned}
(\forall i \in A_x \setminus A_y) \quad & (r_{x,i} \vee \neg r_{z.i}) \wedge (\neg r_{x,i} \vee r_{z,i}) \\
(\forall i \in A_y \setminus A_x) \quad & (r_{y,i} \vee \neg r_{z.i}) \wedge (\neg r_{y,i} \vee r_{z,i}) \\
(\forall i \in A_x \cap A_y) \quad & r_{z,i} \Leftrightarrow (\neg(r_{x,i} \Leftrightarrow r_{y,i}))
\end{aligned}$$

pri čemu se kodiranje gornje ekvivalencije svodi na kodiranje ograničenja $bool\_not\_reif(r_{x,i}, r_{y,i}, r_{z,i})$.

Ograničenje array_set_element

Ovo ograničenje ima oblik:

$$array\_set\_element(var\ int : b,\ array\ [int]\ of\ set\ of\ int : as,\ var\ set\ of\ int : c)$$

i zahteva da važi $as[b] = c$, pri čemu je as niz skupovnih parametara. Ideja kodiranja je sledeća:

$$\bigvee_{i \in A_c \cap I(as)} b = i \wedge c = as[i]$$

pri čemu je $I(as)$ skup indeksa niza as. Jednakost $b = i$ kodira se pomoću dve jedinične klauze:

$$p_{b,i}$$

$$\neg p_{b,i-1}$$

Jednakost $c = as[i]$ kodira se na sledeći način:

$$\begin{aligned}
(\forall i \in A_c \setminus as[i]) \quad & \neg r_{c,i} \\
(\forall i \in A_c \cap as[i]) \quad & r_{c,i}
\end{aligned}$$

Kako bi formula bila u KNF obliku, koriste se pomoćne promenljive. Treba naglasiti da, u slučaju kada je za neko i skup $as[i] \setminus A_c$ neprazan, jednakost $c = as[i]$ ne može biti zadovoljena, pa ne treba dodavati klauze vezane za nju.

Ograničenje array_var_set_element

Ovo ograničenje ima oblik:

$$array\_var\_set\_element(var\ int : b,\ array\ [int]\ of\ var\ set\ of\ int : as,\ var\ set\ of\ int : c)$$

i zahteva da važi $as[b] = c$, pri čemu je as niz skupovnih promenljivih. Kodira se analogno ograničenju $array\_set\_element$, osim što se jednakost $c = as[i]$ ovog puta svodi na kodiranje ograničenja $set\_eq(c, as[i])$.

Ograničenje set_card

Ovo ograničenje ima oblik:

$$set\_card(varsetofint : S, varint : x)$$

i zahteva da važi $x = |S|$. Kako se skupovne promenljive kodiraju pomoću niza iskaznih promenljivih, za kodiranje ovog ograničenja može poslužiti ograničenje lin_bool_eq. Kako zahtevamo da fiksiran broj iskaznih promenljivih iz pomenutog niza uzme vrednost tačno, kodiranje se može vršiti analogno kodiranju ograničenja lin_bool_eq kod koga su svi elementi niza as jednaki 1, a niz bs je pomenuti niz iskaznih promenljivih. Pošto je x celobrojna promenljiva, treba uzeti u obzir sve njene moguće vrednosti:

$$\bigvee_{i \in D(x)} x = i \quad \wedge \quad |S| = i$$

Jednakost $x = i$ kodira se pomoću dve jedinične klauze:

$$p_{x,i}$$

$$\neg p_{x,i-1}$$

dok se jednakost $|S| = i$ kodira pomenutim postupkom svođenja na ograničenje lin_bool_eq.

Kako rezultujuća formula nije u KNF obliku, za svođenje na KNF koriste se pomoćne promenljive.

Ograničenje set_le

Ovo ograničenje ima oblik:

$$set\_le(var\ set\ of\ int : x,\ var\ set\ of\ int : y)$$

i zahteva da važi $x \leq y$, pri čemu se koristi leksikografsko uređenje nad sortiranim listama elemenata skupova. Koraci u kodiranju su sledeći:

- Određuju se l i u koji redom predstavljaju minimum, odnosno maksimum skupa $A_x \cup A_y$
- Kreira se niz xb iskaznih promenljivih sa indeksima od l do u, pri čemu je element $xb[i]$ tačan akko je vrednost i uključena u skup x. Analogno se kreira niz yb.
- Uvode se pomoćne celobrojne promenljive x_{min}, y_{min} i x_{max}, y_{max} koje redom predstavljaju minimalne, odnosno maksimalne vrednosti iz skupova x i y. Ovo se radi analogno kodiranju ograničenja $array\_int\_min$ i $array\_int\_max$.
- Kreira se niz b sa indeksima od l do u. Njegove vrednosti definišu se na sledeći način:
  - Ako je $xb[i] = yb[i]$, tada je $b[i] \Leftrightarrow b[i+1]$. Na ovaj način se, idući sa leva na desno (od manjih ka većim elementima), dokle god se skupovi po tim elementima ne razlikuju poređenje svodi rekurzivno na sledeći element $b[i+1]$. Kodiranju navedenog uslova odgovara sledeća formula:

    $$(q_{xb[i]} \vee q_{yb[i]}) \wedge (\neg q_{xb[i]} \vee \neg q_{yb[i]}) \vee (q_{b[i]} \wedge q_{b[i+1]}) \vee (\neg q_{b[i]} \wedge \neg q_{b[i+1]})$$

    koja se jednostavno može svesti na KNF oblik korišćenjem pomoćnih promenljivih.
  - Ako je $yb[i] = 1$, a $xb[i] = 0$, tada je $b[i] = 1$ akko je $x_{max} < i$. Na ovaj način se kodira sledeća činjenica - x je leksikografski manje ili jednako y ako je i-ti element prvi na kome se skupovi razlikuju, a svi elementi skupa x su manji od i; u suprotnom x neće biti manje ili jednako y. Kodiranju navedenog uslova odgovara sledeća formula:

$$(q_{b[i]} \vee \neg p_{y_{max},i}) \wedge (\neg q_{b[i]} \vee p_{y_{max},i})$$

koja se jednostavno može svesti na KNF oblik.

– Slično prethodnom uslovu, ako je $yb[i] = 0$, a $xb[i] = 1$, tada je $b[i] = 1$ akko je $y_{max} > i$. Kodiranju navedenog uslova odgovara sledeća formula:

$$(q_{b[i]} \vee p_{x_{max},i-1}) \wedge (\neg q_{b[i]} \vee \neg p_{x_{max},i-1})$$

koja se jednostavno može svesti na KNF oblik.

– Specijalno, ako je $i = u$, tada se kodira uslov $b[u] \Leftrightarrow (xb[u] \Rightarrow yb[u])$, analogno ograničenju $bool\_le\_reif(xb[u], yb[u], b[u])$.

- Nakon što je niz b definisan na navedeni način, promenljiva $b[l]$ sadrži informaciju da li važi $x \leq y$. Kako bi se obezbedilo da taj uslov bude ispunjen, dodaje se jedinična klauza $q_{b[l]}$.

Ograničenje set_le_reif

Ovo ograničenje ima oblik:

$$set\_le\_reif(var\ set\ of\ int : x,\ var\ set\ of\ int : y)$$

i predstavlja ekvivalencijski oblik ograničenja set_le. Kodira se analogno navedenmo ograničenju, osim što se na kraju umesto jedinične klauze $q_{b[l]}$ kodira ograničenje $bool\_eq(b[l], r)$.

Ograničenje set_lt

Ovo ograničenje ima oblik:

$$set\_lt(var\ set\ of\ int : x,\ var\ set\ of\ int : y)$$

i zahteva da važi $x < y$, pri čemu se koristi leksikografsko uređenje nad sortiranim listama elemenata skupova. Kodira se analogno ograničenju set_le, osim što se u specijalnom slučaju kada je $i = u$ koristi ograničenje $bool\_lt\_reif(xb[u], yb[u], b[u])$.

Ograničenje set_lt_reif

Ovo ograničenje ima oblik:

$$set\_lt\_reif(var\ set\ of\ int : x,\ var\ set\ of\ int : y)$$

i predstavlja ekvivalencijski oblik ograničenja set_lt. Kodira se analogno navedenom ograničenju, osim što se na kraju umesto jedinične klauze $q_{b[l]}$ kodira ograničenje $bool\_eq(b[l], r)$.

Ekvivalencijski oblici ograničenja

Ograničenja: set_eq_reif, set_in_reif, set_ne_reif, set_subset_reif, set_superset_reif, kao i njihove verzije sa implikacijom, kodiraju se ranije opisanim postupkom kodiranja ekvivalencijskih, odnosno implikacijskih oblika ograničenja.

3.6 Primer kodiranja FlatZinc modela

Kako bi se ilustrovao proces kodiranja konkretnog FlatZinc modela, u ovom odeljku je predstavljen primer kodiranja sledećeg modela:

```
1  var 1..3: x;
2  var 1..3: y;
3  var 1..3: z;
4
5  var bool: a;
6  var bool: b;
7  var bool: c;
8
9  var set of 1..3: s;
10 var set of 1..3: t;
11 var set of 1..3: u;
12
13 constraint int_ne(x, y);
14 constraint int_plus(y, z, 4);
15 constraint int_lin_le([2, 3], [x, z], 7);
16
17 constraint bool_or(a, b, true);
18 constraint array_bool_xor([a, b, c]);
19 constraint bool_le(b, c);
20
21 constraint set_in(x, s);
22 constraint set_subset_reif(u, t, a);
23 constraint set_intersect(s, u, t);
24
25 solve satisfy;
```

Za početak, treba kodirati domene promenljivih. Eksplicitno je potrebno dodati klauze jedino za celobrojne promenljive:

$$\forall i \in \{1, \ldots, 3\}, \quad \neg p_{v,i-1} \vee p_{v,i}$$

$$\neg p_{v,0}$$

$$p_{v,3}$$

za svako $v \in \{x, y, z\}$. Dalje, potrebno je kodirati jedno po jedno ograničenje.

Ograničenje $int\_ne(x, y)$ kodira se po sledećem principu:

$$\bigwedge_{a+b=-2} (\neg h_1 \vee p_{x,a} \vee \neg p_{y,-b-1}) \tag{3.10}$$

$$\bigwedge_{a+b=-2} (\neg h_2 \vee \neg p_{x,-a-1} \vee p_{y,b}) \tag{3.11}$$

$$h_1 \vee h_2 \tag{3.12}$$

pri čemu u formuli (3.10) važi $a \in \{0, \ldots, 3\}, b \in \{-4, \ldots, -1\}$, dok u formuli (3.11) važi $a \in \{-4, \ldots, -1\}, b \in \{0, \ldots, 3\}$.

Ograničenje $int\_plus(y, z, 4)$ kao treći argument ima konstantu 4. Kako je na tom mestu očekivana celobrojna promenljiva, potrebno je kodirati celobrojnu pomoćnu promenljivu c sa fiksiranim domenom jednakim $\{4\}$:

$$\neg p_{c,3}$$

$$p_{c,4}$$

Nakon toga, za kodiranje samog ograničenja koriste se sledeće formule:

$$\bigwedge_{i+j+k=-2} (\neg p_{c,-i-1} \vee p_{y,j} \vee p_{z,k}) \tag{3.13}$$

$$\bigwedge_{i+j+k=-2} (p_{c,i} \vee \neg p_{y,-j-1} \vee \neg p_{z,-b-1}) \tag{3.14}$$

pri čemu u formuli (3.13) važi $i \in \{-5, -4\}, j \in \{0, \ldots, 3\}, k \in \{0, \ldots, 3\}$, dok u formuli (3.14) važi $i \in \{3, 4\}, j \in \{-4, \ldots, -1\}, k \in \{-4, \ldots, -1\}$.

Za kodiranje ograničenja $int\_lin\_le([2, 3], [x, z], 7)$ najpre je neophodno uvesti pomoćnu promenljivu s čija je svrha kodiranje supstitucije $s = 2 \cdot x + 3 \cdot z$. Granice domena promenljive s su $l(s) = 2 \cdot l(x) + 3 \cdot l(z) = 5$ i $u(s) = 2 \cdot u(x) + 3 \cdot u(z) = 15$. Kodiranje supstitucije ide slično kao u prethodnom ograničenju:

$$\bigwedge_{i+j+k=-2} (\neg p_{s,-i-1} \vee p_{x,j} \vee p_{z,k}) \tag{3.15}$$

$$\bigwedge_{i+j+k=-2} (p_{s,i} \vee \neg p_{x,-j-1} \vee \neg p_{z,-b-1}) \tag{3.16}$$

pri čemu u formuli (3.15) važi $i \in \{-16, \ldots, -5\}, j \in \{0, \ldots, 3\}, k \in \{0, \ldots, 3\}$, dok u formuli (3.16) važi $i \in \{4, \ldots, 15\}, j \in \{-4, \ldots, -1\}, k \in \{-4, \ldots, -1\}$.

Kako bi se ograničenje kodiralo, ostaje još da se doda sledeća jedinična klauza:

$$p_{s,7}$$

Kodiranje ograničenje $bool\_or(a, b, true)$ zahteva najpre uvođenje pomoćne promenljive $aorb$ kojom će biti predstavljena konstanta $true$. Ovo se postiže dodavanjem jedinične klauze q_{aorb}. Nakon toga, ograničenje se kodira sledećom formulom:

$$(\neg q_a \vee q_{aorb}) \wedge (\neg q_b \vee q_{aorb}) \wedge (q_a \vee q_b \vee \neg q_{aorb})$$

Ograničenje $array\_bool\_xor([a, b, c])$ zahteva najpre uvođenje pomoćne promenljive axb radi kodiranja supstitucije $axb = a \oplus b$. Pomenuta supstitucija se kodira sledećom formulom:

$$(q_a \vee \neg q_{axb}) \wedge (q_b \vee \neg q_{axb}) \wedge (\neg q_a \vee \neg q_b \vee q_{axb})$$

Sada je dovoljno obezbediti da promenljive axb i c imaju različite vrednosti:

$$(q_{axb} \vee q_c) \wedge (\neg q_{axb} \vee \neg q_c)$$

Ograničenje $int\_le(b, c)$ kodira se jednostavno sledećom klauzom:

$$\neg q_b \vee q_c$$

Ograničenje $set\_in(x, s)$ kodira se sledećim formulama:

$$\bigwedge_{i=1}^{3} (\neg p_{x,i-1} \vee \neg h_i) \wedge (\neg p_{x,i-1} \vee \neg h_i) \wedge (r_{s,i} \vee \neg h_i)$$

$$h_1 \vee h_2 \vee h_3$$

Ograničenje $set\_subset\_reif(u, t, a)$ kodira se sledećim formulama:

$$\neg q_a \vee \neg H_1$$

$$\bigwedge_{i=1}^{3} (r_{u,i} \vee \neg h_i) \wedge (\neg r_{t,i} \vee \neg h_i)$$

$$h_1 \vee h_2 \vee h_3 \vee \neg H_1$$

$$q_a \vee \neg H_2$$

$$\bigwedge_{i=1}^{3} \neg r_{u,i} \vee r_{t,i} \vee \neg H_2$$

$$H_1 \vee H_2$$

Ograničenje $set\_intersect(s, u, t)$ kodira se sledećom formulom:

$$\bigwedge_{i=1}^{3} (r_s \vee \neg r_t) \wedge (r_u \vee \neg r_t) \wedge (\neg r_s \vee \neg r_u \vee r_t)$$

Sveukupno, formula dobijena kodiranjem celog modela ima 51 promenljivu i 203 klauze.

Glava 4

Implementacija

U ovom poglavlju predstavljena je implementacija alata *FlatToSAT*[1] pisanog u jeziku C++ čiji razvoj je centralna tema rada. Ukratko je izložen proces parsiranja FlatZinc modela, a zatim je dat opis najbitnijih delova klase `Encoder` koja je ključna za funkcionisanje alata, kao i opis toka izvršavanja programa.

Parser

Parser jezika FlatZinc generisan je pomoću alata Flex i Bison [6]. Korišćena je gramatika data u okviru formalne specifikacije jezika FlatZinc [14]. Za svaki od osnovnih koncepata jezika (parametri, promenljive, ograničenja, cilj modela) deklarisana je po jedna struktura koja čuva njegove ključne podatke. Takođe, pomoću C++ šablona `variant`, deklarisan je tip unije pomenutih struktura pod nazivom `Item`. Polja struktura su popunjavana korišćenjem akcija nad odgovarajućim pravilima gramatike, a zatim su same strukture dodavane u niz čiji su elementi tipa `Item`, pod nazivom `parsing_result`. Kompletan model je po završetku sačuvan u pomenutom nizu koji je dalje prosleđen klasi `Encoder`.

Klasa Encoder

Glavni deo implementacije alata sadržan je u okviru klase `Encoder`. Javni deo klase čine sledeći metodi:

- konstruktor klase - prima pomenuti niz `parsing_result`, kao i nisku koja označava putanju na kojoj treba napraviti datoteku `formula.cnf` u koju će

[1] `https://github.com/Lojovic/FlatZincToSATConverter`

biti upisana KNF formula

- `encode_to_cnf` - prevodi FlatZinc model u KNF formulu korišćenjem pomoćnih metoda opisanih u nastavku
- `write_to_file` - ispisuje KNF formulu u DIMACS formatu u datoteku `formula.cnf`
- `run_minisat` - pokreće SAT rešavač minisat nad formulom sačuvanom u datoteci `formula.cnf`; izlaz je sačuvan u datoteci `model.out`
- `read_minisat_output` - čita izlaz SAT rešavača iz datoteke `model.out`, a zatim ispisuje poruku o zadovoljivosti (`UNSAT, SAT`). U slučaju zadovoljive formule, izlaz SAT rešavača se dekodira i ispisuju se vrednosti dodeljene promenljivama

U privatnom delu klase najznačajnija su sledeća polja:

- `cnf_clauses` - niz čiji su elementi nizovi literala. Svaki element predstavlja klauzu, a celokupni niz predstavlja KNF formulu. Literali su predstavljeni strukturom koja čuva tip (`enum` sa elementima `ORDER, BOOL_VARIABLE, HELPER, DIRECT, SET_ELEM`), identifikator (ceo broj; vezuje literal za promenljivu iz originalnog problema), pol (tipa `bool`; `false` ukoliko je literal negiran, a u suprotnom `true`) i vrednost za koju je literal vezan (ceo broj; značajno kod literala tipa `ORDER, DIRECT i SET_ELEM`)
- `id_map` - mapa koja određuje promenljivu iz originalnog problema na osnovu identifikatora. Promenljive koje se ne nalaze u ovoj mapi ne obrađuju se prilikom dekodiranja rešenja SAT rešavača (pomoćne promenljive ili one za koje je već dedukovana vrednost)
- `parameter_map, variable_map, array_map` - određuju o kojem se parametru/promenljivoj/nizu radi na osnovu imena. Koriste se prilikom dohvatanja argumenata ograničenja u metodu `encode_constraint`
- `literal_to_num, num_to_literal` - određuju preslikavanje između skupa literala i skupa DIMACS brojeva. Koriste se u metodu `write_clauses_to_file`, kao i prilikom dekodiranja izlaza SAT rešavača

Najznačajniji privatni metodi su:

- `write_clauses_to_file` - ispisuje KNF formulu sačuvanu u polju `cnf_clauses` u pomoćnu datoteku `helper2.cnf`, pritom nadovezujući se na njen sadržaj, a zatim prazni niz `cnf_clauses`. Za razliku od metoda `write_to_file` koji ispisuje KNF formulu koja predstvalja kompletan FlatZinc model, ovaj metod se poziva nakon svakog kodiranja ograničenja radi uštede RAM-a. Takođe, vodi evidenciju o broju klauza i literala, što metod `write_to_file` kasnije koristi kako bi popunio datoteku `helper1.cnf` DIMACS zaglavljem formule (datoteka `formula.cnf` se dobija nadovezivanjem datoteka `helper1.cnf` i `helper2.cnf`)
- `encode_variable` - kodira domen promenljive koja se prosleđuje metodu, na način opisan u prethodnom poglavlju. Popunjava mape `id_map, variable_map` i `array_map`
- `encode_parameter` - popunjava mapu `parameter_map`
- `encode_constraint` - prosleđuje mu se ograničenje čije argumente dohvata pomoću gorepomenutih mapa, a zatim poziva odgovarajuću funkciju za kodiranje
- metodi oblika `encode_ime_ogranicenja` - kodiraju ograničenje sa odgovarajućim imenom na način opisan u prethodnom poglavlju

Tok izvršavanja

Tok izvršavanja programa tipično izgleda ovako:

- Program se pokreće naredbom `./flatzinc_to_sat [path/to/input.fzn]`, pri čemu ukoliko se putanja do FlatZinc modela ne navede očekuje se da će model biti unet preko standardnog ulaza
- Funkcija `main` poziva funkciju `yyparse` koja parsira FlatZinc model i popunjava niz `parsing_result`
- Kreira se instanca klase `Encoder` kojoj se prosleđuje `parsing_result`, a zatim se poziva metod `encode_to_cnf`
- Metod `encode_to_cnf` prolazi kroz niz `parsing_result`, proverava tip trenutnog elementa (parametar, promenljiva, ograničenje) i poziva odgovarajući metod (`encode_parameter, encode_variable, encode_constraint`)

- Nakon izvršavanja metoda `encode_to_cnf`, pozivaju se redom metodi `write_to_file`, `run_minisat` i `read_minisat_output`. Rezultat je ispis informacija o zadovoljivosti problema i eventualnoj zadovoljavajućoj dodeli vrednosti promenljivama na standardni izlaz.

Glava 5

Evaluacija

Evaluacija performansi alata vršena je na računaru sa 16GB RAM-a, Intel Core i5-8350U procesorom sa 4 jezgra (8 niti) i frekvencijom 1.70GHz. Za evaluaciju su korišćeni problemi iz korpusa *minizinc-benchmarks*[1], kao i pojedini problemi obrađeni u okviru specijalnog kursa *Simboličko izračunavanje*[2] održanog na Matematičkom fakultetu tokom školske 2022/23 godine. Parametri su birani kao uzorak skupa ponuđenih parametara u slučaju korpusa *minizinc-benchmarks*, odnosno procenom autora u slučaju problema sa kursa *Simboličko izračunavanje*. Kako su neki od problema inicijalno bili optimizacione prirode, najpre je bilo potrebno rešiti ih pomoću nekog od rešavača koji podržavaju probleme te vrste, a zatim dodati optimizovani izraz sa dobijenom vrednošću kao parametar novog modela koji će za cilj imati zadovoljenje. Poređenje je vršeno sa rešavačem *chuffed* zasnovanom na lenjom generisanju klauza, a rezultati su prezentovani u tabelama $5.1 - 5.17$ datim u nastavku. Simbol / označava da je rešavaču ponestalo memorije, dok simbol + označava da je izvršavanje prekinuto nakon naznačenog vremenskog perioda (najšešće posle proizvoljno izabranih 3 sata i 30 minuta označenih sa 3h30m+).

[1] `https://github.com/MiniZinc/minizinc-benchmarks`

[2] `https://github.com/milanbankovic/symbolic_computing/tree/main/programiranje_ogranicenja/primeri`

Tabela 5.1: queens

n	**FlatToSAT**	**Chuffed**
4	0m0.021s	0m0.079s
8	0m0.106s	0m0.073s
20	0m4.041s	0m0.332s
50	4m37.031s	3h30m+
100	/	3h30m+
200	/	3h30m+
400	/	3h30m+

Tabela 5.2: knights

n, m	**FlatToSAT**	**Chuffed**
8, 4	22m19.342s	0m0.080s
8, 10	38m51.031s	0m0.144s
8, 12	40m9.736s	0m0.120s
8, 14	45m5.703s	0m0.139s

Tabela 5.3: latin-squares

n	**FlatToSAT**	**Chuffed**
3	0.057s	0m0.087s
7	0.661s	0m0.103s
10	3.589s	0m0.109s
12	9.173s	0m0.145s
15	31.587s	0m0.256s
20	2m44.736s	0m0.637s
25	10m36.418s	0m2.85s

Tabela 5.4: schur_numbers

n, c	**FlatToSAT**	**Chuffed**
5, 3	0m0.077s	0m0.071s
7, 3	0m0.083s	0m0.087s
10, 3	0m0.129s	0m0.088s

Tabela 5.5: kakuro

Type	**FlatToSAT**	**Chuffed**
6x6, easy	0m0.210s	0m0.075s
6x6, hard	0m0.228s	0m0.093s
6x6, super	0m0.236s	0m0.087s
8x8, easy	0m0.263s	0m0.093s
8x8, hard	0m0.390s	0m0.078s
8x8, super	0m0.489s	0m0.080s

Tabela 5.6: golomb

n, k	**FlatToSAT**	**Chuffed**
6, 4	0m0.100s	0m0.085s
7, 4	0m0.132s	0m0.089s
17, 6	0m2.998s	0m0.095s
25, 7	0m20.071s	0m0.076s
34, 8	1m51.700s	0m0.577s
44, 9	8m49.740s	0m3.378s

Tabela 5.7: knapsack

n, C, value	**FlatToSAT**	**Chuffed**
6, 10, 35	0m0.046s	0m0.062s
5, 20, 18	0m0.157s	0m0.075s
3, 50, 220	0m0.375s	0m0.089s
8, 15, 165	0m0.436s	0m0.073s
15, 35, 235	0m2.546s	0m0.107s
20, 50, 101	0m3.497s	0m0.073s

Tabela 5.8: allinterval

Instanca	**FlatToSAT**	**Chuffed**
easy1	0m0.674s	0m0.088s
easy2	0m1.695s	0m0.168s
medium1	0m2.486s	0m0.450s
medium2	0m4.730s	0m1.655s
hard1	0m7.980s	0m11.635s
hard2	0m13.568s	0m4.294s

Tabela 5.9: langford

n	FlatToSAT	Chuffed
3	0m0.158s	0m0.082s
4	0m0.387s	0m0.078s
5	0m0.916s	0m0.084s
7	0m3.516s	0m0.111s
8	0m6.149s	0m0.116s
11	0m24.189s	0m0.187s

Tabela 5.10: golfers

m, n	FlatToSAT	Chuffed
2, 2	0m0.946s	0m0.105s
2, 3	1m2.802s	0m0.107s
4, 3	/	0m17.674s

Tabela 5.11: fillomino

Dimenzije	FlatToSAT	Chuffed
3x3	0m0.844s	0m0.085s
4x4	0m5.494s	0m0.116s
4x4	0m7.638s	0m0.353s
5x5	0m17.064s	0m0.124s
5x5	0m19.525s	0m0.132s
5x5	0m19.070s	0m0.183s

Tabela 5.12: nonogram

Instanca	FlatToSAT	Chuffed
non_fast_1	12m25.945s	0m4.152s
non_fast_2	10m44.750s	0m4.875s
non_med_1	9m11.369s	0m12.857s
non_med_2	25m18.027s	1m2.659s
non_awful_1	13m15.544s	46m49.961s
non_awful_2	9m56.723s	0m16.344s

Tabela 5.13: nurse scheduling problem (nsp)

Instanca	**FlatToSAT**	**Chuffed**
14_1	0m2.187s	0m6.217s
14_2	0m1.938s	3h26m22s
14_3	0m1.631s	3h30m+
28_1	0m3.300s	2m7.000s
28_2	0m3.253s	57m38.000s
28_3	0m3.463s	2h29m3s

Tabela 5.14: grid-coloring

n, m	**FlatToSAT**	**Chuffed**
4, 8	0m0.543s	0m0.616s
5, 6	0m0.456s	0m0.124s
7, 8	0m1.318s	0m5.821s
10, 10	0m24.432s	10h+
12, 13	10h+	10h+

Tabela 5.15: steiner-triples

n	**FlatToSAT**	**Chuffed**
3	0m0.014s	0m0.092s
7	0m0.632s	0m0.140s
9	0m2.091s	0m0.515s
13	0m15.246s	8m20.000s
15	1m25.902s	3h30m+
19	22m7.875s	3h30m+

Tabela 5.16: equation solving (EQ)

Instanca	**FlatToSAT**	**Chuffed**
eq20	/	0m0.095s

Tabela 5.17: slow_convergence

n	**FlatToSAT**	**Chuffed**
100	/	0m0.562s

Može se primetiti da je alat uspeo da reši većinu problema na kojima je vršena evaluacija. Takođe, kod većine problema, može se opaziti prirodan trend porasta

vremena izvršavanja sa povećanjem vrednosti parametara. Za nekoliko problema kod kojih to nije slučaj, parametri su kalibrisani tako da se težina instance određuje prema standardnim CSP rešavačima, što ne oslikava težinu rešavanja za pristup koji koristi alat FlatToSAT. Kod nekoliko problema (*queens*, *nsp*, *grid-coloring*, *steiner-triples*), alat je uspeo da reši instance koje *chuffed* nije uspeo da reši u posmatranom vremenskom periodu. Postoji nekoliko problema kod kojih je alat imao problem sa memorijom, što se može pripisati veličini KNF formule koja se prosleđuje SAT rešavaču. Takvi problemi su najčešće imali promenljive sa velikim domenima (veličina ≥ 1000) ili su prilikom kodiranja linearnih ograničenja nastajale pomoćne promenljive sa velikim domenima. Treba naglasiti da su dobijena rešenja proverena, pri čemu nisu uočene greške, što povećava sigurnost u ispravnost alata (iako ona nije formalno dokazana, pa je moguće da greške postoje).

Glava 6

Zaključak

U ovom radu dat je detaljan opis postupka svođenja FlatZinc modela na KNF formulu. Najpre su uvedeni određeni pomoćni koncepti (poput kodiranja primitivnih zbirova i razlika, kao i korišćenja pomoćnih promenljivih), a zatim je opisan pristup kodiranju domena promenljivih različitih tipova, kao i pojedinačnih FlatZinc ograničenja. Predstavljena je implementacija alata napisanog u jeziku C++ koji korišćenjem pomenutog pristupa prevodi FlatZinc model u KNF formulu u DIMACS formatu, dok za rešavanje dobijene formule koristi minisat SAT rešavač. Evaluacija alata vršena je poređenjem vremena izvršavanja sa *chuffed* CSP rešavačem na 17 problema. Pokazalo se da alat uspeva da nađe rešenja čak i za teške instance većine problema, ali kod određenog broja problema nailazi na prepreku u vidu memorijskih ograničenja.

Uprkos dobrim rezultatima evaluacije, alat je razvijen sa primarnim ciljem funkcionalnosti, a ne efikasnosti. Samim tim, jedan mogući pravac unapređenja uključivao bi optimizaciju postojeće implementacije, korišćenje drugih načina kodiranja pojedinačnih ograničenja, kao i drugih šema kodiranja domena promenljivih. Drugi mogući smer istraživanja mogao bi da podrazumeva formalizaciju postupka kodiranja pojedinačnih ograničenja, kako bi se stekla sigurnost u ispravnost alata. Najzad, u eri munjevitog napretka različitih tehnika mašinskog učenja, nameće se mogućnost integracije pomenutih tehnika sa alatom, u cilju odabira optimalnog načina kodiranja određenog problema.

Bibliografija

[1] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 151–158. ACM, 1971.

[2] Robert M. Haralick and Gordon L. Elliott. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14(3):263–313, 1980.

[3] Jix. Dimacs format documentation, 2025.

[4] Martin Kay. The proper place of men and machines in language translation. *Machine Translation*, 12:3–23, 1980.

[5] J. L. Lassez, M. J. Maher, and K. Marriott. Unification revisited. In Mauro Boscarol, Luigia Carlucci Aiello, and Giorgio Levi, editors, *Foundations of Logic and Functional Programming*, pages 67–113, Berlin, Heidelberg, 1988. Springer Berlin Heidelberg.

[6] John Levine. *Flex & Bison: Text Processing Tools*. O'Reilly Media, Sebastopol, CA, 2009.

[7] Alan K. Macworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[8] Joao Marques-Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning sat solvers. In *Handbook of satisfiability*, pages 133–182. ios Press, 2021.

[9] Filip Marić Mirko Stojadinović. mesat: Multiple encodings of csp to sat. *Constraints*, 19(4):380–403, 2014.

[10] Nicholas Nethercote, Peter J Stuckey, Ralph Becket, Sebastian Brand, Gregory J Duck, and Guido Tack. Minizinc: Towards a standard cp modelling lan-

guage. In *International Conference on Principles and Practice of Constraint Programming*, pages 529–543. Springer, 2007.

[11] Olga Ohrimenko, Peter J Stuckey, and Michael Codish. Propagation via lazy clause generation. *Constraints*, 14:357–391, 2009.

[12] Naoyuki Tamura, Akiko Taga, Satoshi Kitagawa, and Mutsunori Banbara. Compiling finite linear csp into sat. volume 14, 09 2006.

[13] Chuffed Development Team. Chuffed: A solver for constraint satisfaction problems, 2025.

[14] MiniZinc Development Team. *FlatZinc Specification*, 2024.

[15] MiniZinc Development Team. *MiniZinc Handbook*, 2024.

[16] MiniZinc Development Team. Geas: A constraint solver for minizinc, 2025.

[17] Sugar Development Team. *Sugar: A CSP to SAT Translator*, 2025.

[18] Toby Walsh. Sat v csp. In Rina Dechter, editor, *Principles and Practice of Constraint Programming – CP 2000*, pages 441–456, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

Biografija autora

Matija Lojović rođen je 11.06.1999. u Aranđelovcu. U rodnom gradu je završio osnovnu školu „Ilija Garašanin" 2014. godine i gimnaziju „Miloš Savković", prirodno-matematički smer, 2018. godine, ponevši Vukovu diplomu iz obe. Po završetku srednje škole, upisao je osnovne akademske studije na Matematičkom fakultetu Univerziteta u Beogradu, smer Informatika. Osnovne studije je završio 2023. godine sa prosekom 9,33 i time stekao zvanje Diplomirani informatičar. Po završetku osnovnih, upisao je master studije na istom fakultetu, takođe na smeru Informatika. Od 2023. godine je zaposlen na Matematičkom fakultetu, kao saradnik u nastavi na Katedri za računarstvo i informatiku.