Kapitola 3

Modelovanie úloh vo výrokovej logike

Použitie splniteľnosti

Cieľom tejto časti je ukázať, ako je možné využiť výrokovú logiku pre riešenie úloh. V zásade je hľadanie riešenia úlohy založené na úlohe splniteľnosti – samotné riešenie je reprezentované vhodnou interpretáciou logických symbolov. Postup riešenia sa skladá z troch etáp:

- reprezentácia úlohy pomocou vety vo výrokovej logike,
- prekódovanie logickej vety,
- hľadanie modelu logickej vety.

Prvá etapa je vlastne modelovaním riešeného problému pomocou výrazových prostriedkov, ponúkaných výrokovou logikou (kap. 1) – teda pomocou viet nad množinou logických symbolov, ktoré reprezentujú väzbu na riešenú úlohu. Výsledok modelovania má podobu logickej vety, ktorá nie je syntakticky obmedzená na niektorý špeciálny tvar.

Druhá etapa je hľadaním takej interpretácie použitých symbolov, ktorá má za následok pravdivú interpretáciu tej logickej vety, ktorá reprezentuje celý riešený problém. Pre tento účel sa používa moderný SAT solver založený na využití prehľadávacích metód (kap. 2), ktorý určuje splniteľnosť alebo nesplniteľnosť logických viet a v prípade ich splniteľnosti je schopný nájsť aj modely týchto viet.

Medzi obe spomenuté hlavné etapy je ešte potrebné vložiť prekódovanie reprezentácie úlohy z formy všeobecnej logickej vety cez logickú vetu v CNF tvare do príslušného (typicky *Dimacs*) formátu, ktorý je akceptovaný použitým solverom. Prípadne je aj možné logickú vetu zjednodušiť, ak

ILUSTRAČNÝ PRÍKLAD

Opäť sa stretávame so študentmi Ferom, Ondrom, Stanom, Rudom a Jarom z príkladu Pr. 1.1. Tentokrát sa pripravujú na turnaj, na ktorý tvrdo trénovali.
Úloha 1: *Je potrebné naplánovať priebeh turnaja aby sa mohol stretnúť každý súťažiaci s každým zo svojich súperov, pričom každá dvojica sa má stretnúť iba raz. Vzhľadom na počet súťažiacich turnaj bude mať päť kôl, pričom v každom kole budú súťažiť dve dvojice a jeden zo študentov bude mať voľno.*
Úloha 2: *Turnaj skončil a už je iba spomienkou. O výsledkoch, dosiahnutých v turnaji, jednotliví súťažiaci povedali:*

- *Stano: turnaj vyhral Jaro a Fero bol druhý*
- *Fero: pokiaľ si pamätám, Jaro skončil lepšie ako Ondro*
- *Ondro: turnaj vyhral Rudo a Jaro bol druhý*
- *Rudo: Stano nebol posledný a Jaro bol podpriemerný.*
- *Jaro: Stano a Ondro povedali spolovice pravdu a spolovice nepravdu.*

Aké boli konečné výsledky turnaja ak každý skončil na inom mieste?
Úloha 3: *Chlapci na turnaj cestovali každý s desať kilogramovou batožinou. Cestou boli nútení použiť výťah, ktorý mal nosnosť obmedzenú na 180 kg. Každý sa viezol so svojou batožinou alebo svoju batožinu zveril kamarátovi, ktorý sa potom viezol so svojou a jednou cudzou batožinou. Ako sa mali zorganizovať aby im stačili tri jazdy výťahom, ak ich hmotnosti boli: Fero – 96 kg, Ondro – 84 kg, Stano – 79 kg, Rudo – 72 kg a Jaro – 87 kg ?*
Reprezentácia:
Artefakty – študenti, kolá, dvojice, umiestnenia, batožiny a jazdy:

$$V_s = \{Fero, Ondro, Stano, Rudo, Jaro\} \qquad V_k = \{1,2,3,4,5\}$$
$$V_d = \{1,2\} \qquad V_m = \{1,2,3,4,5\}$$
$$V_b = \{batF, batO, batS, batR, batJ\} \qquad V_j = \{1,2,3\}$$

Vytvorenie rozpisu: 50 premenných $\mathcal{V}_{k,s,d}$ (k – kolo, s – študent, d – dvojica)
Určenie poradia: 25 premenných $\mathcal{V}_{s,m}$ (s – študent, m – umiestnenie)
Cestovanie výťahom: 30 premenných $\mathcal{V}_{o,j}$ (o – študent alebo batožina, j – jazda) a 25 premenných $\mathcal{W}_{b,s}$ (b – batožina, s – študent)

Riešenie 1: *Jedným z možných rozpisov turnaja je: 1. kolo: Fero a Ondro, Stano a Rudo, 2. kolo: Fero a Stano, Ondro a Jaro, 3. kolo: Ondro a Rudo, Fero a Jaro, 4. kolo: Fero a Rudo, Stano a Jaro, 5. kolo: Ondro a Stano, Rudo a Jaro.*
Riešenie 2: *Celkové poradie: 1. Rudo, 2. Fero, 3. Stano, 4. Jaro, 5. Ondro.*
Riešenie 3: *Jeden z možných spôsobov použitia výťahu je: 1. jazda: Jaro bez batožiny a Stano so svojou batožinou, 2. jazda: Ondro a Rudo s batožinami, 3. jazda: Fero so svojou a Jarovou batožinou.*

Ilustr. 3.1: Príklad pre modelovanie pomocou výrokovej logiky

toto nerobí solver v rámci predspracovania svojho vstupu pred začiatkom hľadania vhodnej interpretácie.

Samotné nájdenie reprezentácie úlohy pozostáva z niekoľkých krokov, ktoré sú znázornené algoritmom Alg. 3.1.

ALGORITMUS MODELOVANIA ÚLOHY

vstup: úloha U v ľubovoľnom (napr. slovnom) tvare
výstup: logická veta F reprezentujúca úlohu

1. $(\mathcal{V}, \mathcal{D}, \mathcal{C}) := \textit{návrh\_premenných\_a\_podmienok}(\ U\ )$
2. $(S_1, F_1) := \textit{kódovanie\_premenných}(\ \mathcal{V}\ )$
3. $(S_2, F_2) := \textit{kódovanie\_podmienok}(\ \mathcal{C}\ )$
4. $(S, F) := \textit{úprava\_na\_CNF}(\ S_1 \cup S_2,\ F_1 \wedge F_2\ )$

Alg. 3.1: Reprezentácia úlohy pomocou CNF

Hľadanie vhodnej interpretácie logickej vety v CNF tvare je možné považovať vlastne za riešenie špeciálneho prípadu *úlohy s ohraničeniami* nad enumeračnými doménami. Pretože logické symboly majú dvojhodnotové domény $\{TRUE, FALSE\}$, nad hodnotami ktorých sa neuvažuje usporiadanie a klauzuly s n literálmi ($n = 1, 2, \ldots$) sú n-árne ohraničenia nad použitými symbolmi, tak možno pomerne priamočiaro úlohy s ohraničeniami nad enumeračnými doménami mapovať na logické symboly a klauzuly zostavené z literálov použitých symbolov.

Preto je potrebné najprv úlohu v prvom kroku formalizovať ako úlohu s ohraničeniami – teda pomocou konečného počtu premenných $\mathcal{V}$, konečných domén $\mathcal{D}$ týchto premenných daných pomocou vymenovania jednotlivých prvkov, a ohraničení (podmienok) $\mathcal{C}$, ktoré musia spĺňať kombinácie hodnôt jednotlivých premenných.

V druhom kroku je potrebné definované premenné $\mathcal{V}$ prekódovať – každú premennú $\mathcal{V}_i$, ktorá má doménu $\mathcal{D}_i$ s n prvkami ($n \geq 2$), je potrebné vyjadriť jedným alebo niekoľkými symbolmi (každý s binárnou doménou) a ohraničeniami nad týmito symbolmi, odlišujúcimi prípustné a neprípustné kombinácie hodnôt symbolov. Výsledkom tohto kroku je množina logických symbolov S_1 a množina klauzúl F_1 využívajúca tieto symboly (ktorá môže byť prázdna).

Nasledujúci krok zabezpečí kódovanie ohraničení $\mathcal{C}$ – každé ohraničenie $\mathcal{C}_i$ nad podmnožinou premenných je potrebné vyjadriť ako jedno alebo viac ohraničení nad množinou symbolov S_1. Pri tejto operácii môže vzniknúť potreba dodefinovať ďalšie pomocné symboly. Výsledkom je množina pomocných symbolov S_2 (môže byť aj prázdna) a množina klauzúl, definovaných nad množinou symbolov $S_1 \cup S_2$.

Posledným krokom je spojenie doteraz získaných výsledkov prechádzajúcich krokov a ich transformácia do výsledného tvaru, ktorým je často logická veta reprezentovaná vo formáte CNF.

NÁVRH PREMENNÝCH

Voľba premenných je pomerne intuitívna aktivita, pre ktorú je možné poskytnúť zopár doporučení avšak nie je možné ju vyjadriť vo forme algoritmu. Jedným z možných postupov je v riešenej úlohe identifikovať množiny artefaktov (objekty, udalosti, vlastnosti, atď.). A z týchto množín následne vytvárať premenné reprezentujúce prvky Kartézskeho súčinu vybraných množín. Premenné je možné vytvárať

- kombinovaním prvkov množín, v najjednoduchšom prípade z každej množiny sa vyberie po jednom prvku,
- kombinovaním prvkov vybraných z jednej množiny.

Samozrejme je možné oba prístupy spájať – z niektorých množín vyberať po jednom prvku, z niektorých po viac prvkov a z niektorých nevyberať vôbec.

KÓDOVANIE PREMENNÝCH S DVOJPRVKOVÝMI DOMÉNAMI

Každá takáto premenná reprezentuje nejakú kombináciu prvkov z vybraných množín a svojou hodnotou určuje, či daná kombinácia je prípustná alebo nie. Keďže sa jedná o premenné s dvojprvkovými doménami, ich kódovanie pomocou symbolov (množina S_1) a klauzúl nad týmito symbolmi (množina F_1) je triviálne. Jednoducho každá premenná bude reprezentovaná samostatným symbolom, ktorý v prípade prípustnosti reprezentovanej kombinácie prvkov bude interpretovaný ako pravdivý a v prípade neprípustnosti reprezentovanej kombinácie prvkov bude zase interpretovaný ako nepravdivý. Kvôli kódovaniu premenných nie je potrebné definovať žiadne dodatočné ohraničenia (prípad prázdnej množiny klauzúl F_1).

Príklad 3.1 Pre ilustráciu uvažujme ilustračný problém na strane 68. Vo všetkých častiach vystupuje množina študentov $V_s = \{$Fero, Ondro, Stano, Rudo, Jaro$\}$, k tomu v prvej časti pristupujú množina kôl $V_k = \{1, 2, 3, 4, 5\}$ a množina dvojíc $V_d = \{1, 2\}$, v druhej časti zase množina umiestnení $V_m = \{1, 2, 3, 4, 5\}$ a v tretej časti množina batožín $V_b = \{$batF, batO, batS, batR, batJ$\}$ a množina jázd $V_j = \{1, 2, 3\}$.

Aplikovaním prvého princípu kombinovania prvkov množín by pre plánovanie turnaja bolo možné kombinovať prvky množín V_s, V_k a V_d a vytvoriť 50 premenných typu $\mathcal{V}_{k,s,d}$, kde index k reprezentuje prvok z množiny kôl, s prvok z množiny študentov a d prvok z množiny dvojíc, Takto napríklad premenná $\mathcal{V}_{3,F,2}$ reprezentuje to, že Fero v treťom kole je členom druhej dvojice (ak symbol kódujúci túto premennú je pravdivý) resp. že Fero v treťom kole nie je členom druhej dvojice (ak kódujúci symbol nie je pravdivý).

Podobne pre určenie výsledku turnaja je možné kombinovaním prvkov množín V_s a V_m vytvoriť 25 premenných typu $\mathcal{V}_{s,m}$, kde s reprezentuje prvok z množiny študentov a m zase prvok z množiny umiestnení. Teraz napríklad premenná $\mathcal{V}_{F,2}$ reprezentuje to, že Fero skončil na druhom mieste (ak kódujúci symbol je pravdivý) resp. že Fero neskončil na druhom mieste (ak kódujúci symbol je nepravdivý).

Podobne pre plánovanie cesty výťahom je možné využiť kombinovanie prvkov z rôznych množín. Možno vytvoriť dva typy premenných. Jeden typ obsahuje 30 premenných tvaru $\mathcal{V}_{o,j}$ pre o reprezentujúci objekt zo zjednotenia množín študentov a ich batožín a j zastupujúci jazdu výťahu, kde napríklad premenná $\mathcal{V}_{O,2}$ reprezentuje Ondrovu účasť v druhej jazde výťahu. Druhý typ obsahuje 25 premenných tvaru $\mathcal{W}_{b,s}$, kde s reprezentuje batožinu a s zase študenta, prepravujúceho túto batožinu, pričom napríklad premenná $\mathcal{W}_{batF,O}$ vyjadruje, že Ferovu batožinu prepravuje Ondro.

Aplikovaním druhého princípu vzájomného kombinovania prvkov jednej množiny by pre plánovanie turnaja bolo možné vytvoriť 100 premenných typu $\mathcal{V}_{k,s1,s2}$ ($s_1 \neq s_2$) pre k reprezentujúci prvok z množiny kôl a s_1 ako aj s_2 vyjadrujúce prvky z množiny študentov, kde napríklad premenná $\mathcal{V}_{2,R,J}$ reprezentuje to, že v druhom kole sa v jednej dvojici stretli Rudo a Jaro (ak príslušný symbol je pravdivý) resp. že sa tí dvaja v druhom kole nestretli (ak príslušný kódujúci symbol je nepravdivý).

Podobne kombinovaním prvkov z jednej množiny je možné pre výsledok turnaja vytvoriť 20 premenných typu $\mathcal{V}_{s1,s2}$ ($s_1 \neq s_2$) s s_1 aj s_2 reprezentujúcimi prvky z množiny študentov, kde napríklad premenná $\mathcal{V}_{O,S}$ hovorí, že Ondro sa umiestnil lepšie ako Stano resp. že Ondro sa neumiestnil lepšie ako Stano (v závislosti na interpretácii kódujúceho symbolu).

•

Voľba typu premenných

Je teda zrejmé, že pre tú istú úlohu je možné vytvoriť viacero rôznych reprezentácií. Pre plánovanie turnaja jedna reprezentácia používala explicitné označenie dvojice jej pomenovaním zatiaľ čo druhá reprezentácia dvojicu vyjadrovala implicitne vymenovaním jej členov. Pre zistenie poradia študentov jedna reprezentácia stavala na absolútnom poradí študentov, zatiaľ čo druhá používala relatívne poradie študentov. Preferencia medzi

reprezentáciami môže byť založená na tom, v ktorej reprezentácii sa jednoduchšie vyjadria podmienky dané úlohou alebo ktorá reprezentácia sa ukáže vhodnejšia pre použitý SAT solver.

Príklad 3.2 Podmienka viazaná na číslo dvojice (napr. Ondro súťaží stále v prvej dvojici) sa dá jednoducho vyjadriť pri explicitnej reprezentácii dvojice $(\neg\mathcal{V}_{1,O,2}, \ldots, \neg\mathcal{V}_{5,O,2})$ avšak nedá sa vyjadriť pri implicitnej reprezentácii dvojíc.

Podmienka viazaná na kombinovanie študentov (napr. Stano a Jaro súťažia v párnom kole) sa dá vyjadriť v oboch reprezentáciách, avšak toto vyjadrenie je pri implicitnej reprezentácii dvojíc $(\mathcal{V}_{2,S,J} \vee \mathcal{V}_{4,S,J} \vee \mathcal{V}_{2,J,S} \vee \mathcal{V}_{4,J,S})$ jednoduchšie ako v prípade explicitnej reprezentácii dvojíc $((\mathcal{V}_{2,S,1} \wedge \mathcal{V}_{2,J,1}) \vee (\mathcal{V}_{2,S,2} \wedge \mathcal{V}_{2,J,2}) \vee (\mathcal{V}_{4,S,1} \wedge \mathcal{V}_{4,J,1}) \vee (\mathcal{V}_{4,S,2} \wedge \mathcal{V}_{4,J,2}))$.

Absolútne umiestnenie (napr. Jaro skončil na treťom mieste) sa jednoduchšie vyjadrí pri absolútnom poradí $(\mathcal{V}_{J,3})$ avšak dá sa vyjadriť aj pri relatívnom poradí $(=_2 (\mathcal{V}_{J,F}, \mathcal{V}_{J,O}, \mathcal{V}_{J,S}, \mathcal{V}_{J,R}))$.

Relatívne umiestnenie (napr. Jaro skončil lepšie ako Ondro) je triviálne vyjadriť pri relatívnom poradí $(\mathcal{V}_{J,O})$ avšak dá sa, hoci aj zložitejšie, vyjadriť aj pri absolútnom poradí $(\mathcal{V}_{J,1} \rightarrow \mathcal{V}_{O,2} \vee \mathcal{V}_{O,3} \vee \mathcal{V}_{O,4} \vee \mathcal{V}_{O,5}, \ldots, \mathcal{V}_{J,4} \rightarrow \mathcal{V}_{O,5})$. •

NÁVRH PODMIENOK

Medzi navrhnutými premennými môžu existovať závislosti, určujúce vzájomné vzťahy (kompatibilitu či nekompatibilitu) medzi kombináciami prvkov množín, reprezentovaných týmito premennými. Tieto vzťahy sú úlohovo špecifické, pretože sú založené na informáciách týkajúcich sa priamo požiadaviek riešených úloh. Keďže za validné riešenie sa považuje iba to, ktoré spĺňa všetky požiadavky kladené naň definíciou úlohy, opomenutie explicitného vyjadrenia niektorej podmienky môže mať za následok generovanie nevalidných riešení.

Kvôli následnému kódovaniu podmienok je vhodné, ak podmienky sú presne špecifikované, najlepšie nielen popisným spôsobom ale použitím nejakého formálneho zápisu.

Príklad 3.3 Uvažujme prípad plánovania turnaja s použitím premenných typu $\mathcal{V}_{k,s,d}$, kde $k \in V_k$, $s \in V_s$ a $d \in V_d$. Tieto premenné sú navzájom závislé, napríklad nie je možné aby dve premenné $\mathcal{V}_{3,F,2}$ a $\mathcal{V}_{3,F,1}$ mohli súčasne vyjadrovať prípustnú situáciu (kompatibilitu dvoch kombinácií prvkov použitých množín), pretože Fero nemôže byť naraz na dvoch miestach.

Pre zabezpečenie vzájomných závislostí medzi premennými je potrebné

dodržať podmienky[1]

- každý študent v každom kole môže byť maximálne v jednej dvojici (buď v jednej ak v danom kole súťaží alebo v žiadnej ak v danom kole má voľno)
  $\forall s \in V_s, \forall k \in V_k : \quad \leq_1 (\mathcal{V}_{k,s,1}, \mathcal{V}_{k,s,2})$,
- každý študent počas turnaja musí byť presne v štyroch dvojiciach (súťaží so štyrmi súpermi)
  $\forall s \in V_s : \quad =_4 (\mathcal{V}_{1,s,1}, \mathcal{V}_{1,s,2}, \mathcal{V}_{2,s,1}, \mathcal{V}_{2,s,2}, \ldots, \mathcal{V}_{5,s,1}, \mathcal{V}_{5,s,2})$,
- každá dvojica spája presne dvoch rôznych študentov (nikto nesúťaží sám so sebou ale vždy s jedným zo svojich protivníkov)
  $\forall k \in V_k, \forall d \in V_d : \quad =_2 (\mathcal{V}_{k,F,d}, \mathcal{V}_{k,O,d}, \mathcal{V}_{k,S,d}, \mathcal{V}_{k,R,d}, \mathcal{V}_{k,J,d})$,
- žiadna dvojica študentov sa nemôže opakovať v rôznych kolách (so žiadnym protivníkom sa nesúťaží viackrát)
  $\forall s_1, s_2 \in V_s$ kde $s_1 \neq s_2, \forall k_1, k_2 \in V_k$ kde $k_1 \neq k_2, \forall d_1, d_2 \in V_d$:
  $\mathcal{V}_{k_1,s_1,d_1} \wedge \mathcal{V}_{k_1,s_2,d_1} \rightarrow \neg(\mathcal{V}_{k_2,s_1,d_2} \wedge \mathcal{V}_{k_2,s_2,d_2})$.
- každý študent súťaží v štyroch kolách (pretože má štyroch súperov)
  $\forall s \in V_s : \quad =_4 (\mathcal{V}_{1,s,1} \vee \mathcal{V}_{1,s,2}, \mathcal{V}_{2,s,1} \vee \mathcal{V}_{2,s,2}, \ldots, \mathcal{V}_{5,s,1} \vee \mathcal{V}_{5,s,2})$
- každý študent s každým protivníkom súťaží práve raz
  $\forall s_1, s_2 \in V_s$ kde $s_1 \neq s_2$:
  $=_1 (\mathcal{V}_{1,s_1,1} \wedge \mathcal{V}_{1,s_2,1}, \mathcal{V}_{1,s_1,2} \wedge \mathcal{V}_{1,s_2,2}, \ldots, \mathcal{V}_{5,s_1,d_2} \wedge \mathcal{V}_{5,s_2,2})$

Okrem uvedených podmienok je možné definovať aj ďalšie podmienky, napríklad že každá dvojica sa musí vyskytnúť aspoň v jednom kole. •

REDUNDANCIA PODMIENOK

Podmienka môže byť *redundantná* alebo *neredundantná*. Redundancia podmienky vyjadruje vzťah danej podmienky k ostatným definovaným podmienkam. Je založená na tom, že zabezpečenie platnosti niektorých podmienok môže znamenať súčasne aj zabezpečenie platnosti iných podmienok. Redundantnou je teda taká podmienka, ktorej platnosť vyplýva z iných podmienok v rámci uvažovanej množiny podmienok. Jedná sa o relatívnu vlastnosť – podmienka v nejakej množine podmienok môže byť redundantnou zatiaľ čo v inej množine podmienok redundantnou nemusí byť.

Redundantné podmienky môžu byť vypustené z množiny podmienok. Tým sa zjednoduší vytváraný model úlohy a jeho výsledná podoba v tvare CNF bude jednoduchšia (v zmysle počtu klauzúl). Na druhej strane to

[1] $\leq_k$, $=_k$ a $\geq_k$ sú kardinalitné ohraničenia, reprezentujúce podmienku, že maximálne, presne alebo minimálne k alternatív z daného zoznamu alternatív je prípustných.

nemusí znamenať rýchlejšie nájdenie riešenia solverom – často solver dokáže pracovať efektívnejšie v prípade, že má k dispozícii viac klauzúl (pri hľadaní nesprávnym smerom dokáže skôr dospieť ku konfliktu), ktoré musí hľadanou interpretáciou symbolov splniť. Keďže to však neplatí všeobecne, nedá sa poskytnúť všeobecné doporučenie, či redundantné podmienky vynechávať alebo nie.

MINIMÁLNA MNOŽINA PODMIENOK

Pri definovaní podmienok v skutočnosti stačí použiť *minimálnu množinu podmienok* (taká množina, ktorá obsahuje iba neredundantné klauzuly). Vypustenie hociktorej podmienky z tejto minimálnej množiny má za následok generovanie síce formálne správnych avšak vzhľadom na riešenú úlohu nevalidných riešení – po vynechaní niektorej podmienky z tejto minimálnej množiny už totiž nie je garantované dodržanie všetkých závislostí medzi premennými.

Príklad 3.4 V predchádzajúcom príklade šiestich podmienok napríklad podmienka "každý študent v každom kole môže byť maximálne v jednej dvojici" vyplýva z platnosti podmienok "každý študent počas turnaja musí byť presne v štyroch dvojiciach" a "každý študent súťaží v štyroch kolách", pretože nie je možné zostaviť taký rozvrh, ktorý by prvú z týchto podmienok nespĺňal ale ostatné dve áno. Podobne podmienky "každý študent s každým protivníkom súťaží práve raz" a "každá dvojica spája presne dvoch rôznych študentov" dokážu nahradiť podmienku "žiadna dvojica študentov sa nemôže opakovať v rôznych kolách".

Úlohu minimálnej množiny podmienok môžu plniť napríklad tieto tri podmienky:

- každý študent v každom kole môže byť maximálne v jednej dvojici,
- každá dvojica spája presne dvoch rôznych študentov,
- žiadna dvojica študentov sa nemôže opakovať v rôznych kolách.

Neuvažovanie prvej podmienky by mohlo mať za následok vznik plánu 1. kolo: Fero-Ondro a Fero-Rudo, 2.kolo: Fero-Jaro a Fero-Stano, 3. kolo: Ondro-Jaro a Ondro-Stano, 4. kolo: Ondro-Rudo a Stano-Rudo, 5. kolo: Stano-Jaro a Rudo-Jaro.

Nerešpektovanie druhej podmienky by pripustilo plán tvaru 1. kolo: Fero- a Stano-, 2.kolo: Fero- a Stano-, 3. kolo: Ondro- a Stano-, 4. kolo: Rudo- a Stano-, 5. kolo: Jaro- a Rudo-.

Vypustenie poslednej podmienky by umožnilo rozvrh typu 1. kolo: Fero-Ondro a Stano-Jaro, 2.kolo: Fero-Rudo a Ondro-Stano, 3. kolo: Fero-Jaro a Stano-Rudo, 4. kolo: Fero-Ondro a Rudo-Jaro, 5. kolo: Ondro-Jaro a Stano-Rudo.

•

Kardinalitné ohraničenia

Často je potrebné vyjadriť (ako už aj bolo potrebné použiť v predchádzajúcich príkladoch podmienok) nejaké obmedzenie na platnosť iba určitého počtu alternatív z nejakej množiny dostupných alternatív[2]. Formálne to je možné vyjadriť

$<_k (X_1, X_2, \ldots, X_n)$ – menej ako k alternatív z n môže byť pravdivých a teda ostatné z danej množiny musia byť nepravdivé,

$\leq_k (X_1, X_2, \ldots, X_n)$ – maximálne k alternatív z n môže byť pravdivých (teda vrátane počtu rovného hodnote k)

$>_k (X_1, X_2, \ldots, X_n)$ – viac ako k alternatív z n musí byť pravdivých,

$\geq_k (X_1, X_2, \ldots, X_n)$ – minimálne k alternatív z n musí byť pravdivých,

$=_k (X_1, X_2, \ldots, X_n)$ – práve k alternatív z n je pravdivých.

Náhrada $<_k$ $>_k$

Podmienka "menej ako" môže byť nahradená podmienkou "maximálne" použitím ich vzájomného vzťahu

$$<_k (X_1, X_2, \ldots, X_n) \quad = \quad \leq_{k-1} (X_1, X_2, \ldots, X_n)$$

a analogickým spôsobom je možné tiež nahradiť podmienku "viac ako" pomocou podmienky "minimálne". Teda nie je potrebné sa špeciálne venovať podmienkam ostrých nerovností.

Náhrada $=_k$

Keďže podmienka "práve" môže byť vyjadrená ako súčasná platnosť ostatných dvoch podmienok založených na neostrých nerovnostiach

$$=_k (X_1, X_2, \ldots, X_n) \; = \; \leq_k (X_1, X_2, \ldots, X_n) \wedge \; \geq_k (X_1, X_2, \ldots, X_n)$$

tak ani rovnosť nie je nutné uvažovať osobitným spôsobom.

Náhrada $\geq_k$

A navyše keďže podmienky "aspoň" a "najviac" sú ekvivalentné podľa vzťahu

$$\geq_k (X_1, X_2, \ldots, X_n) \quad = \quad \leq_{n-k} (\neg X_1, \neg X_2, \ldots, \neg X_n)$$

tak je pri kódovaní kardinalitných ohraničení postačujúce sa zaoberať iba jedným ohraničením – podmienkou "najviac k". Táto platí pre $0 < k < n$ ($k = 0$ môže byť jednoducho reprezentované pomocou n jednotkových

[2] Alternatívami môžu byť trebárs symboly alebo logické výrazy nad symbolmi alebo vyjadrenie že premenná nadobúda nejakú hodnotu zo svojej domény.

klauzúl tvaru $\neg X_i$, prípad $k = n$ reprezentuje vždy splnené ohraničenie, ktoré možno vynechať).

BINOMICKÉ KÓDOVANIE

Podmienku $\leq_k$ možno kódovať viacerými spôsobmi, z ktorých je najjednoduchším *binomické kódovanie*. Zo sémantiky podmienky vyplýva, že pre nejakú hodnotu k nie je povolené, aby existovala nejaká $(k+1)$-tica alternatív, napríklad $X_1, \ldots, X_k, X_{k+1}$, z ktorých by všetky boli pravdivé – aspoň jedna z nich musí byť nepravdivá. Toto môže byť reprezentované ako

$$\neg(X_1 \wedge \ldots \wedge X_k \wedge X_{k+1}) \equiv \neg X_1 \vee \ldots \vee \neg X_k \vee \neg X_{k+1}$$

čo vylučuje porušenie podmienky platnosti maximálne k alternatív z tejto konkrétnej $(k+1)$-tice. A pretože toto platí pre každú možnú kombináciu $k+1$ alternatív, je potrebné generovať takúto klauzulu pre všetky možné kombinácie dĺžky $k+1$

$$\bigwedge_{M \subseteq \{1,\ldots,n\}, |M|=k+1} \quad \bigvee_{i \in M} \neg X_i$$

čo vyžaduje $\begin{pmatrix} n \\ k+1 \end{pmatrix}$ klauzúl dĺžky $k+1$.

Príklad 3.5 Pre ilustráciu uvažujme opäť plánovanie turnaja s reprezentáciou pomocou premenných typu $\mathcal{V}_{k,s,d}$ pre kombinovanie kôl, študentov a dvojíc. Jednou z podmienok použitia takéhoto vyjadrenia bolo, že každá dvojica spája presne dvoch rôznych študentov, ktorá bola vyjadrená ako kardinalitné ohraničenie tvaru

$$\forall k \in V_k, \forall d \in V_d : \quad =_2 (\mathcal{V}_{k,F,d}, \mathcal{V}_{k,O,d}, \mathcal{V}_{k,S,d}, \mathcal{V}_{k,R,d}, \mathcal{V}_{k,J,d})$$

Keďže takéto premenné majú binárne domény, tak každú z nich je možné reprezentovať priamo jedným symbolom, kde napríklad premenná $\mathcal{V}_{1,F,1}$ bude reprezentovaná symbolom $X_{1,F,1}$. A teda uvedenú podmienku je potrebné zakódovať ako kardinalitné ohraničenie $=_2$ nad množinou piatich symbolov. Tak napríklad pre tretie kolo a druhú skupinu je možné toto zakódovať pomocou dvoch viet, kde prvá veta, kódujúca $\leq_2$, má tvar CNF s desiatimi klauzulami

$$\begin{array}{ll}
\neg X_{3,F,2} \vee \neg X_{3,O,2} \vee \neg X_{3,S,2} & \neg X_{3,F,2} \vee \neg X_{3,O,2} \vee \neg X_{3,R,2} \\
\neg X_{3,F,2} \vee \neg X_{3,O,2} \vee \neg X_{3,J,2} & \neg X_{3,F,2} \vee \neg X_{3,S,2} \vee \neg X_{3,R,2} \\
\neg X_{3,F,2} \vee \neg X_{3,S,2} \vee \neg X_{3,J,2} & \neg X_{3,F,2} \vee \neg X_{3,R,2} \vee \neg X_{3,J,2} \\
\neg X_{3,O,2} \vee \neg X_{3,S,2} \vee \neg X_{3,R,2} & \neg X_{3,O,2} \vee \neg X_{3,S,2} \vee \neg X_{3,J,2} \\
\neg X_{3,O,2} \vee \neg X_{3,R,2} \vee \neg X_{3,J,2} & \neg X_{3,S,2} \vee \neg X_{3,R,2} \vee \neg X_{3,J,2}
\end{array}$$

a druhá veta, kódujúca komplementárne ohraničenie $\geq_2$, má tvar CNF

$$X_{3,F,2} \vee X_{3,O,2} \vee X_{3,S,2} \vee X_{3,R,2} \qquad X_{3,F,2} \vee X_{3,O,2} \vee X_{3,S,2} \vee X_{3,J,2}$$
$$X_{3,F,2} \vee X_{3,O,2} \vee X_{3,R,2} \vee X_{3,J,2} \qquad X_{3,F,2} \vee X_{3,S,2} \vee X_{3,R,2} \vee X_{3,J,2}$$
$$X_{3,O,2} \vee X_{3,S,2} \vee X_{3,R,2} \vee X_{3,J,2}$$

s piatimi klauzulami, každá so štyrmi literálmi. •

Často sa vyskytujúcim prípadom kardinalitného ohraničenia je prípad $=_1 (X_1, \ldots, X_n)$. V tomto prípade binomické kódovanie pre $\geq_1$ je veľmi jednoduché – jedna klauzula spájajúca v disjunkcii všetkých n alternatív. Naopak, kódovanie pre $\leq_1$ je síce tvorené iba dvojliterálovými klauzulami, avšak problematický je ich počet, ktorý je rádovo n^2 a teda s rastúcou hodnotou n ich počet príliš rýchlo rastie.

Kvôli veľkému počtu generovaných klauzúl preto existuje mnoho ďalších kódovaní. Jedným z nich je *binárne* kódovanie, ktoré je založené na logaritmickom kódovaní. Pre $\leq_k (X_1, \ldots, X_n)$ je k pôvodnej n-tici alternatív pridaných k skupín $\lceil log_2 n \rceil$ pomocných symbolov, navzájom rozlišujúcich jednotlivé alternatívy, ktoré umožnia vytvoriť logaritmické kódovanie alternatív. Previazanie pôvodných symbolov na tieto pomocné symboly sa realizuje implikáciami. Najlepší spôsob vysvetlenia tohto kódovania je však pomocou ilustračného príkladu.

BINÁRNE KÓDOVANIE

Príklad 3.6 Pre ilustráciu tohto kódovania pre určenie výsledku turnaja opäť uvažujme s reprezentáciou pomocou premenných typu $\mathcal{V}_{k,s,d}$ pre kombinovanie kôl, študentov a dvojíc. Jednou z podmienok bolo, že každá dvojica spája maximálne dvoch študentov, čo je vyjadrené v tvare

$$\forall k \in V_k, \forall d \in V_d : \quad \leq_2 (\mathcal{V}_{k,F,d}, \mathcal{V}_{k,O,d}, \mathcal{V}_{k,S,d}, \mathcal{V}_{k,R,d}, \mathcal{V}_{k,J,d})$$

Nech opäť sú tieto premenné reprezentované priamo jedným symbolom (premenná $\mathcal{V}_{3,F,2}$ pomocou symbolu $X_{3,F,2}$) a teda uvedenú podmienku je v prípade druhého kola a prvej dvojice potrebné zakódovať ako kardinalitné ohraničenie $\leq_2$ nad množinou piatich symbolov $X_{2,F,1}$, $X_{2,O,1}$, $X_{2,S,1}$, $X_{2,R,1}$ a $X_{2,J,1}$.

Pre vzájomné rozlíšenie piatich alternatív postačujú tri pomocné symboly ($\lceil log_2 5 \rceil = 3$). Keďže v našom ohraničení $\leq_k$ je $k = 2$, je potrebné použiť dve trojice pomocných symbolov. A tak pre druhé kolo a prvú dvojicu je možné ohraničenie zakódovať pomocou sady implikácií

$$X_{2,F,1} \rightarrow \quad (\neg X_a \wedge \neg X_b \wedge \neg X_c) \quad \vee \quad (\neg X_d \wedge \neg X_e \wedge \neg X_f)$$
$$X_{2,O,1} \rightarrow \quad (\neg X_a \wedge \neg X_b \wedge X_c) \quad \vee \quad (\neg X_d \wedge \neg X_e \wedge X_f)$$

Rozšírenie: Kódovanie kardinalitných ohraničení sekvenčným čítačom

SEKVENČNÉ KÓDOVANIE KARDINALITNÝCH OHRANIČENÍ

Okrem binomického a binárneho kódovania sú aj iné kódovania (napr. *sekvenčné*, *commander*, *product*, *bimander*, *kardinalitné siete*, atď.), ktoré podobne ako binárne kódovanie znižujú počet klauzúl oproti použitiu binomického kódovania za cenu definovania dodatočných pomocných symbolov.

Sekvenčné kódovanie je založené na sekvenčnom elektronickom obvode, pozostávajúcom z n registrov. Každý register počíta počet pravdivých vstupov, pričom zohľadňuje pravdivosť "svojho" vstupu a počet pravdivých predošlých vstupov (počítaný prechádzajúcim registrom). Prvý register tak sleduje a zohľadňuje iba jeden vstup, zatiaľ čo posledný register sleduje síce iba jeden vstup avšak zohľadňuje všetky vstupy. Samotné kódovanie je realizované podľa predpisu pravidiel, ktoré pracujú okrem vstupov X_i aj s novodefinovanými pomocnými symbolmi $R_{i,j}$.

Pre ohraničenie $\leq_k (X_1, \ldots, X_n)$ to je päť pravidiel, popisujúcich propagáciu hodnoty FALSE obvodom dopredným smerom:

$$\bigwedge_{i=1}^{n-1} \neg X_i \vee R_{i,1} \qquad \bigwedge_{j=2}^{k} \neg R_{1,j}$$

$$\bigwedge_{i=2}^{n-1} \bigwedge_{j=1}^{k} \neg R_{i-1,j} \vee R_{i,j} \qquad \bigwedge_{i=2}^{n-1} \bigwedge_{j=2}^{k} \neg X_i \vee \neg R_{i-1,j-1} \vee R_{i,j}$$

$$\bigwedge_{i=2}^{n} \neg X_i \vee \neg R_{i-1,k}$$

Celkovo sekvenčné kódovanie vytvára $k(n-1)$ nových pomocných symbolov a generuje $2nk + n - 3k - 1$ klauzúl s jedným až tromi literálmi.

Pre ohraničenie $\geq_k (X_1, \ldots, X_n)$ to je opäť päť pravidiel, pričom však teraz sa vychádza z propagácie hodnoty TRUE obvodom spätným smerom:

$$\neg R_{1,1} \vee X_1 \qquad R_{n,k} \qquad \bigwedge_{i=2}^{n} \bigwedge_{j=2}^{k} \neg R_{i,j} \vee R_{i-1,j-1}$$

$$\bigwedge_{j=2}^{k} \neg R_{1,j} \qquad \bigwedge_{i=2}^{n} \bigwedge_{j=1}^{k} \neg R_{i,j} \vee R_{i-1,j} \vee X_i$$

Teraz sekvenčné kódovanie vytvára kn nových pomocných symbolov a generuje $2nk - n - k + 2$ klauzúl s jedným až tromi literálmi.

$$\begin{aligned} X_{2,S,1} \rightarrow \quad & (\neg X_a \wedge X_b \wedge \neg X_c) \quad \vee \quad (\neg X_d \wedge X_e \wedge \neg X_f) \\ X_{2,R,1} \rightarrow \quad & (\neg X_a \wedge X_b \wedge X_c) \quad \vee \quad (\neg X_d \wedge X_e \wedge X_f) \\ X_{2,J,1} \rightarrow \quad & (X_a \wedge \neg X_b \wedge \neg X_c) \quad \vee \quad (X_d \wedge \neg X_e \wedge \neg X_f) \end{aligned}$$

kde X_a, X_b a X_c je prvá trojica pomocných symbolov a X_d, X_e a X_f je zase druhá trojica pomocných symbolov. Teda jednotlivé alternatívy používajú navzájom rôzne kombinácie pomocných literálov v každej zo skupín týchto pomocných symbolov.

Výber jednej z alternatív spôsobí určitú interpretáciu pomocných symbolov jednej trojice a výber druhej alternatívy nastaví interpretáciu pomocných symbolov v druhej trojici. Tým pádom pri danom nastavení pomocných symbolov nezostáva možnosť výberu žiadnej z ostatných alternatív, pretože ostatné implikácie môžu byť splnené iba v prípade, že ich predpoklad je interpretovaný ako nesplnený. Pri takomto kódovaní je teda zabezpečené, že najviac k alternatív bude zvolených.

Pretože pravidlo $Z \rightarrow (X_1 \wedge \ldots \wedge X_{\lceil log_2 n \rceil}) \vee (Y_1 \wedge \ldots \wedge Y_{\lceil log_2 n \rceil})$ sa dá nahradiť pomocou $\lceil log_2 n \rceil^2$ klauzúl

$$\begin{array}{cc} \neg Z \vee X_1 \vee Y_1 & \neg Z \vee X_1 \vee Y_2 \\ & \ldots \\ \neg Z \vee X_1 \vee Y_{\lceil log_2 n \rceil} & \neg Z \vee X_2 \vee Y_1 \\ & \ldots \\ \neg Z \vee X_{\lceil log_2 n \rceil} \vee Y_{\lceil log_2 n \rceil - 1} & \neg Z \vee X_{\lceil log_2 n \rceil} \vee Y_{\lceil log_2 n \rceil} \end{array}$$

tak celkový počet generovaných klauzúl pre n alternatív ohraničenia $\leq_k$ je $n\lceil log_2 n \rceil^k$. To je menší počet generovaných klauzúl ako pri binomickom kódovaní pre väčšie hodnoty n a malé hodnoty k..

•

Symetria riešení

Prítomnosť symetrie znamená, že reprezentácia problému pripúšťa viacero riešení, ktoré sú navzájom zameniteľné. Ukážkou takejto zameniteľnosti v prípade plánovania turnaja pri použití premenných typu $\mathcal{V}_{k,s,d}$ a ich priameho kódovania je symetria v rámci jedného kola, kde dve riešenia sa môžu líšiť zámenou dvojíc, napríklad

- *riešenie 1*: ..., $X_{1,F,1}$, $X_{1,O,1}$, $X_{1,S,2}$, $X_{1,R,2}$, ...,
- *riešenie 2*: ..., $X_{1,F,2}$, $X_{1,O,2}$, $X_{1,S,1}$, $X_{1,R,1}$, ...,

alebo symetria v rámci turnaja, kde dve riešenia sa môžu líšiť zámenou kôl, napríklad

- *riešenie 1*: ..., $X_{1,F,1}$, $X_{1,O,1}$, $X_{1,S,2}$, $X_{1,R,2}$, ..., $X_{2,F,1}$, $X_{2,S,1}$, $X_{2,O,2}$, $X_{2,J,2}$, ...,

- *riešenie 2*: $\ldots, X_{2,F,1}, X_{2,O,1}, X_{2,S,2}, X_{2,R,2}, \ldots, X_{1,F,1}, X_{1,S,1}, X_{1,O,2}, X_{1,J,2}, \ldots,$

Výsledkom prítomnosti symetrie môže byť sťaženie nájdenia riešenia metódami, ktoré postupne nachádzajú interpretáciu jednotlivých symbolov, pretože pre nejakú skupinu symbolov môžu nájsť interpretáciu vedúcu na jedno riešenie a pre inú skupinu zase interpretáciu vedúcu na iné riešenie – a tieto dve parciálne interpretácie nemusia byť navzájom kompatibilné a nedajú sa navzájom kombinovať (a teda aspoň jednu z tých parciálnych interpretácií je potrebné "odučiť" aby bola šanca nájsť jedno z riešení).

RUŠENIE SYMETRIE

Výskyt symetrických riešení sa považuje skôr za nevýhodu. Preto je vhodné upraviť reprezentáciu problému tak, aby sa čo najviac obmedzil výskyt týchto symetrických riešení. Možným spôsobom, ako obmedzovať symetriu, je využívať usporiadanie symbolov či ich indexov. Tento princíp je reprezentovaný v požadovaní, aby vyprodukované riešenie dodržiavalo nejaké predpísané usporiadanie svojich zložiek – doplnením vhodných ohraničení sa vylúčia tie riešenia, ktoré sú síce z hľadiska riešenia úlohy validné avšak požadované usporiadanie nespĺňajú.

Príklad 3.7 V prípade symbolov typu $X_{k,s,d}$ je možné indexy k a d považovať za usporiadané, keďže ich hodnoty je možné navzájom usporiadať (ich hodnoty sú z intervalu prirodzených čísel). No hodnoty, ktoré môžu byť dosadzované za index s sú z množiny, nad ktorou zatiaľ nie je usporiadanie. Ak sa použije lexikografické zotriedenie, tak sa získa poradie

$$Fero < Jaro < Ondro < Rudo < Stano$$

a teda v rámci tohto usporiadania všetci študenti, ktorí sú $\leq$ ako Ondro vytvárajú podmnožinu troch študentov $\{Fero, Jaro, Ondro\}$. Podobne $Fero + 3$ bude značiť Ruda a $Fero + 0$ bude značiť Fera.

Skupiny v rámci kola potom môžme usporiadať tak, že ten študent zo zúčastnených súťažiacich v danom kole, ktorý je z nich v danom poradí prvý, bude súťažiť vždy v prvej skupine. Keďže v každom kole má voľno práve jeden študent, tak sa táto podmienka týka iba Fera a Jara a je možné ju formálne vyjadriť ako

$$\bigwedge_{k=1}^{5} \bigwedge_{j=0}^{1} \left((\bigwedge_{i=0}^{j-1} \neg X_{k,(F+i),1} \rightarrow \neg X_{k,(F+j),2} \right)$$

pričom pre nejaké konkrétne kolo k sa rozvinie do klauzúl

$$\top \rightarrow \neg X_{k,(F+0),2}$$
$$\neg X_{k,(F+0),1} \rightarrow \neg X_{k,(F+1),2}$$

hovoriacich, že Fero (F+0) nemôže byť v druhej dvojici a taktiež Jaro (F+1) nemôže byť v druhej dvojici ak v prvej dvojici nie je Fero. Ak by bol turnaj organizovaný takým spôsobom, že v nejakom kole môžu mať voľno až traja súťažiaci, stačí v podmienke vyššie nastaviť $j = 0...3$ a rozvoj podmienky by obsahoval aj klauzuly

$$\neg X_{k,(F+0),1} \wedge \neg X_{k,(F+1),1} \rightarrow \neg X_{k,(F+2),2}$$
$$\neg X_{k,(F+0),1} \wedge \neg X_{k,(F+1),1} \wedge \neg X_{k,(F+2),1} \rightarrow \neg X_{k,(F+3),2}$$

stanovujúce, že Ondro nemôže byť v druhej dvojici ak v prvej dvojici nie je Fero ani Jaro a podobne Rudo nemôže byť v druhej dvojici ak v prvej dvojici nie sú Fero, Jaro ani Ondro. •

REDUNDANCIA A SYMETRIA

Ak by boli dvojice v rámci kôl usporiadané, je možné jednotlivé kolá usporiadať napríklad podľa prvej dvojice. Toto by bolo jednoduché, ak by sme napríklad vedeli pozície súťažiacich v rámci dvojíc – primárne by sa triedilo podľa prvej pozície prvej dvojice a sekundárne podľa druhej pozície tejto dvojice. Teda ak by napríklad prvá dvojica v nejakom kole bola $s_1 - s_2$ a prvá dvojica v neskoršom kole bola $s_3 - s_4$, tak by sa požadovala platnosť $s_1 < s_3$ v prípade rôznych súťažiacich na prvej pozícii a platnosť $s_2 < s_4$ v prípade toho istého súťažiaceho na prvej pozícii (teda $s_1 = s_3$).

Dvojica je síce explicitne reprezentované pri type premenných $\mathcal{V}_{k,s,d}$, avšak este chýba reprezentácia pozície. Preto niekedy sa volí reprezentácia, ktorá na prvý pohľad obsahuje redundantné údaje, ktoré však je možné využiť pri rušení symetrie.

Príklad 3.8 Namiesto $\mathcal{V}_{k,s,d}$ je možné zvoliť redundantnú reprezentáciu $\mathcal{V}_{k,s,d,p}$, kde p s doménou $\{1, 2\}$ vyjadruje pozíciu v dvojici d.

Pre dosiahnutie usporiadanie kôl na základe hráčov v prvej dvojici je možné pridať podmienky pre túto redundantnú reprezentáciu

$\forall s_1, s_2 \in V_s$ kde $s_1 < s_2$, $\forall k_1, k_2 \in V_k$ kde $k_1 < k_2$:
$\neg X_{k_1,s_2,1,1} \vee \neg X_{k_2,s_1,1,1}$
$\forall s, s_1, s_2 \in V_s$ kde $s_1 < s_2$, $\forall k_1, k_2 \in V_k$ kde $k_1 < k_2$:
$X_{k_1,s,1,1} \wedge X_{k_2,s,1,1} \rightarrow \neg X_{k_1,s_2,1,2} \vee X_{k_2,s_1,1,2}$

ktoré zabezpečia požadované usporiadanie jednotlivých kôl (tým že vylúčia tie usporiadanie, ktoré nezodpovedajú požiadavke). •

PREBERANIE DOMÉNOVÝCH PREMENNÝCH

Doteraz sme uvažovali iba prípad, keď premenné boli navrhované takým spôsobom, že každá z nich reprezentovala nejakú kombináciu elementov z množín artefaktov, poskytovaných definíciou riešenej úlohy. Domény takýchto premenných boli dvojprvkové – buď reprezentovaná kombinácia elementov bola prípustná alebo nie.

No môže nastať aj taký prípad, že priamo definícia úlohy pracuje s premennými, ktorých doména obsahuje určitý počet prvkov (typicky je doména s viac ako dvomi prvkami). V tomto prípade môže byť účelné nevytvárať premenné ako prvky Kartézskeho súčinu množín ale priamo reflektovať doménové premenné. Samozrejme návrh podmienok, určujúci závislosti medzi premennými, musí zohľadniť takýto spôsob návrhu premenných.

Príklad 3.9 Pre ilustráciu uvažujme druhú časť ilustračného problému na strane 68. V probléme určenia výsledkov turnaja vystupuje množina študentov $V_s = \{\text{Fero, Ondro, Stano, Rudo, Jaro}\}$ a k nej množina umiestnení $V_m = \{1, 2, 3, 4, 5\}$.

Namiesto kombinovania prvkov množín V_s a V_m a vytvorenia 25 premenných typu $\mathcal{V}_{s,m}$, kde s reprezentuje prvok z množiny študentov a m zase prvok z množiny umiestnení, je možné použiť päť premenných $\mathcal{F}$, $\mathcal{O}$, $\mathcal{S}$, $\mathcal{R}$ a $\mathcal{J}$, Všetky tieto premenné majú rovnakú doménu $\{1, 2, 3, 4, 5\}$. Priradenie $\mathcal{F} = 2$ reprezentuje to, že Fero skončil na druhom mieste, priradenie $\mathcal{O} = 3$ hovorí, že tretie miesto obsadil Ondro. Analogicky premenné $\mathcal{S}$, $\mathcal{R}$ a $\mathcal{J}$ reprezentujú umiestnenie Stana, Ruda a Jara.

Pre zabezpečenie vzájomných závislostí medzi premennými je potrebné dodržať podmienky

- každý študent obsadil iné miesto (neuvažuje sa s delením nejakého umiestnenia medzi viacerých účastníkov): *alldiff*$(\mathcal{F}, \mathcal{O}, \mathcal{S}, \mathcal{R}, \mathcal{J})$
- Stanovo tvrdenie (v zmysle Jarovho výroku): $(\mathcal{J} = 1) \oplus (\mathcal{F} = 2)$
- Ferovo tvrdenie: $\mathcal{J} < \mathcal{O}$
- Ondrovo tvrdenie (v zmysle Jarovho výroku): $(\mathcal{R} = 1) \oplus (\mathcal{J} = 2)$
- Rudovo tvrdenie: $(\mathcal{S} \neq 5) \wedge (\mathcal{J} > 3)$

kde "alldiff" reprezentuje ohraničenie, požadujúce aby vymenované premenné nadobudli navzájom rôzne hodnoty (žiadna dvojica z týchto premenných nesmie mať priradenú rovnakú hodnotu). •

KÓDOVANIE PREMENNÝCH

Pri reprezentovaní úloh sa teda možno často stretnúť aj s prípadom, keď v popise problému vystupuje nejaká premenná, ktorej doména je tvorená množinou vymenovaných hodnôt, ktoré premenná môže nadobúdať. Ak je takáto premenná binárna (môže nadobúdať iba jednu z dvoch možných hodnôt), tak takúto premennú je možné reprezentovať priamo symbolom (napríklad premennú $\mathcal{V}$ symbolom X) – ak symbol platí, tak premenná

nadobudla jednu z hodnôt, a ak symbol neplatí, tak premenná nadobudla druhú hodnotu.

Kódovací problém nastáva, ak doména premennej $\mathcal{V}$ je tvorená množinou možných hodnôt $\{H_1, H_2, \ldots, H_n\}$, pričom $n > 2$. Vtedy nie je možné triviálne kódovanie tejto premennej jedným symbolom, ale je potrebné použiť symbolov viac.

PRIAME KÓDOVANIE PREMENNEJ

Pri *priamom* kódovaní sa použije n symbolov, pričom každý reprezentuje jednu z hodnôt – symbol Y_i bude reprezentovať skutočnosť, že premenná $\mathcal{V}$ nadobudla i-tu hodnotu H_i zo svojej domény ($\mathcal{V} = H_i$).

Ku kódovaniu je potom ešte potrebné pridať podmienku, že premenná musí nadobudnúť práve jednu hodnotu.

$$\leq_1 (Y_1, Y_2, Y_3, \ldots, Y_n) \quad \wedge \quad \geq_1 (Y_1, Y_2, Y_3, \ldots, Y_n)$$

Podmienka $\leq_1$ bude pri metóde binomického kódovania kardinalitných ohraničení kódovaná v tvare

$$(\neg Y_1 \vee \neg Y_2) \wedge (\neg Y_1 \vee \neg Y_3) \wedge \ldots \wedge (\neg Y_{n-1} \vee \neg Y_n)$$

a pri binárnom kódovaní kardinalitných ohraničení zase v tvare

$$(Y_1 \rightarrow \neg X_1 \wedge \ldots \wedge \neg X_{\lceil log_2 n \rceil}) \wedge (Y_2 \rightarrow \ldots) \wedge \ldots \wedge (Y_n \rightarrow \ldots)$$

kde $\{X_1, \ldots, X_{\lceil log_2 n \rceil}\}$ sú pomocné symboly doplnené binárnym kódovaním. Komplementárna podmienka $\geq_1$ bude pri metóde binomického kódovania kardinalitných ohraničení kódovaná vetou

$$(Y_1 \vee \ldots \vee Y_n)$$

a vzhľadom na jednoduchosť tohto kódovania sa v prípade tejto podmienky binárne kódovanie nepoužíva.

Symboly $Y_1, \ldots . Y_n$ budú súčasťou množiny symbolov S_1 a kardinalitné ohraničenie zase súčasťou množiny klauzúl F_1 (krok kódovanie premenných v algoritme Alg. 3.1).

Príklad 3.10 V príklade určovania výsledkov turnaja by bolo potrebné použiť 25 symbolov $F_1 \ldots F_5$, $O_1 \ldots O_5$, $S_1 \ldots S_5$, $R_1 \ldots R_5$ a $J_1 \ldots J_5$. Symboly $F_1 \ldots F_5$ reprezentujú možné priradenia hodnôt premennej $\mathcal{F}$ – napríklad symbol F_2 reprezentuje $\mathcal{F} = 2$. Ostatné symboly analogickým spôsobom reprezentujú priradenia hodnôt zvyšným premenným.

Definícia symbolov musí byť ešte doplnená piatimi kardinalitnými ohraničeniami

$$\forall X, X = F, O, S, R, J \quad : \quad =_1 (X_1, X_2, X_3, X_4, X_5)$$

ktoré je potrebné transformovať na desať ohraničení typu $\leq_1$.

LOGARITMICKÉ KÓDOVANIE PREMENNEJ

Pri *logaritmickom kódovaní* je potrebných pre jednu premennú použiť $m = \lceil log_2 n \rceil$ symbolov. Vzťah medzi symbolmi, použitými pre reprezentáciu hodnôt domény premennej a samotnými hodnotami je zložitejší. Pre zjednodušenie uvažujme konkrétny príklad domény s piatimi hodnotami H_1 až H_5. Pre ich kódovanie sú potrebné tri symboly X_1 až X_3 podľa nasledujúcej tabuľky:

	X_1	X_2	X_3
H_1	FALSE	FALSE	FALSE
H_2	FALSE	FALSE	TRUE
H_3	FALSE	TRUE	FALSE
H_4	FALSE	TRUE	TRUE
H_5	TRUE	FALSE	FALSE

	X_1	X_2	X_3
H_1	FALSE	FALSE	FALSE
H_2	FALSE	FALSE	TRUE
H_3	FALSE	TRUE	
H_4	TRUE	FALSE	
H_5	TRUE	TRUE	

Tab. 3.1: Ukážka logaritmického kódovania

Fakt, že premenná nadobudla nejakú hodnotu, sa vyjadrí ako konjunkcia literálov. Buď sa použije vždy všetkých m symbolov (neoptimalizovaná podoba – ľavá tabuľka v Tab. 3.1) alebo pre niektoré hodnoty sa použije m a pre iné zase $m - 1$ symbolov (optimalizovaná podoba – pravá tabuľka v Tab. 3.1). Teda napríklad použitie hodnoty H_4 sa vyjadrí buď ako $\neg X_1 \wedge X_2 \wedge X_3$ alebo ako $X_1 \wedge \neg X_2$.

Výhodou logaritmického kódovania je menší počet potrebných symbolov. To, že premenná nesmie nadobudnúť súčasne dve a viac hodnôt, je už zabezpečené samotným kódovaním. Pri neoptimalizovanej podobe je nutné ešte kódovať podmienku, že premenná musí nadobudnúť aspoň jednu hodnotu (teda že napríklad $X_1 \wedge X_2 \wedge X_3$ nie je prípustné). To môže byť kódované vetou

$$((\neg X_1 \wedge \ldots \wedge \neg X_{\lceil log_2 n \rceil}) \vee (\neg X_1 \wedge \ldots \wedge X_{\lceil log_2 n \rceil})) \vee \ldots$$

zahŕňajúcou všetky tie konjunkcie symbolov, ktoré reprezentujú možné hodnoty premennej. Toto by bolo súčasťou množiny klauzúl F_1 (krok kódovanie premenných v algoritme Alg. 3.1). Pri optimalizovanej podobe však nie je potrebné kódovať žiadnu dodatočnú podmienku a kódovanie ničím neprispieva do množiny klauzúl F_1.

Príklad 3.11 Vzhľadom k tomu, že $\lceil log_2 5 \rceil = 3$, tak v príklade určovania výsledkov turnaja by bolo potrebné použiť 15 symbolov $F_1 \ldots F_3, O_1 \ldots O_3$,

$S_1 \dots S_3$, $R_1 \dots R_3$ a $J_1 \dots J_3$. Symboly $F_1 \dots F_3$ konjunkciou svojich literálov reprezentujú možné priradenia hodnôt premennej $\mathcal{F}$ podľa

$$\begin{aligned}
\mathcal{F} = 1 &: \neg F_1 \wedge \neg F_2 \wedge \neg F_3 \\
\mathcal{F} = 2 &: \neg F_1 \wedge \neg F_2 \wedge F_3 \\
\mathcal{F} = 3 &: \neg F_1 \wedge F_2 \\
\mathcal{F} = 4 &: F_1 \wedge \neg F_2 \\
\mathcal{F} = 5 &: F_1 \wedge F_2
\end{aligned}$$

pričom je zrejmé, že bol použitý optimalizovaný variant kódovania. Ostatné symboly analogickým spôsobom reprezentujú priradenia hodnôt zvyšným premenným.

Definíciu symbolov nie je potrebné doplniť žiadnym dodatočným ohraničením. •

Sekvenčné kódovanie premennej

Pri kódovaní *sekvenčným čítačom* je pre jednu premennú potrebné použiť $n-1$ symbolov $X_1, \dots, X_{n-1}$. To, že premenná nadobudla i-tu hodnotu zo svojej domény sa vyjadrí tak, že symboly $X_1, \dots, X_{i-1}$ budú pravdivé, zatiaľ čo ostatné symboly budú nepravdivé. Pre validnosť takéhoto kódovania je potrebné zabezpečiť

- ak symbol X_i je pravdivý, tak všetky symboly $X_j, j < i$ musia byť tiež pravdivé,
- ak symbol X_i je nepravdivý, tak všetky symboly $X_j, j > i$ musia byť tiež nepravdivé.

Toto je možné reprezentovať pomocou podmienky

$$\bigwedge_{i=1}^{n-2} (X_i \vee \neg X_{i+1})$$

vylučujúcej prípad $\neg X_i \wedge X_{i+1}$ keď po nepravdivom symbole nasleduje pravdivý. Táto podmienka obohatí množinu klauzúl F_1 (krok kódovanie premenných v algoritme Alg. 3.1). Dôsledkom existencie tejto podmienky je to, že kódovaná premenná vždy nadobúda práve jednnu hodnotu a neexistuje spôsob ako premennej nepriradiť žiadnu hodnotu alebo naopak hodnôt viacero.

Napríklad pre doménu s piatimi hodnotami sa podmienka rozvinie do klauzulárneho tvaru

$$(X_1 \vee \neg X_2) \wedge (X_2 \vee \neg X_3) \wedge (X_3 \vee \neg X_4)$$

Priradenie $\mathcal{V} = H_i$ by teda bolo reprezentované konjunkciou

$$X_1 \wedge \ldots \wedge X_{i-1} \wedge \neg X_i \wedge \ldots \wedge X_{n-1}$$

avšak stačí vyjadriť interpretáciu iba dvoch symbolov v tvare $H_{i-1} \wedge \neg H_i$ a uvedená podmienka zabezpečí správnu interpretáciu ostatných symbolov. Dokonca v prípade $\mathcal{V} = H_1$ stačí $\neg X_1$ a pre prípad $\mathcal{V} = H_n$ stačí X_{n-1}.

Výhodou tohto typu kódovanie je, že umožňuje reprezentovať nielen priradenie jednej konkrétnej hodnoty premennej ale aj to, že premenná nadobúda hodnotu z nejakého intervalu $i, \ldots . j$ – stačí nastaviť $X^I_{i-1} = TRUE$ a $X^I_j = FALSE$ (pomocou výrazu $X_{i-1} \wedge \neg X_j$).

Vďaka tomu napríklad vyjadrenie nerovnosti $\mathcal{V} \leq H_3$ pri doméne s piatimi hodnotami stačí kódovať pomocou $\neg X_2$. Vďaka podmienke obohacujúcej množinu klauzúl F_1 symboly X_3 a X_4 budú interpretované ako nepravdivé a to znamená že kódovaná premenná nemôže nadobudnúť hodnotu H_4 ani H_5. Symboly X_1 a X_2 zatiaľ interpretované nebudú – čím sa ponecháva možnosť neskôr zúžiť množinu hodnôt $\{H_1, H_2, H_3\}$ na jednu z nich. Analogicky pri kódovaní nerovnosti $\mathcal{V} \geq H_3$ sa vystačí s X_2. Teraz vďaka podmienke na kódovanie bude symbol X_1 interpretovaný ako pravdivý a to znamená že kódovaná premenná nemôže nadobudnúť hodnotu H_1 ani H_2.

Príklad 3.12 V príklade určovania výsledkov turnaja by bolo potrebné použiť 20 symbolov $F_1 \ldots F_4, O_1 \ldots O_4, S_1 \ldots S_4, R_1 \ldots R_4$ a $J_1 \ldots J_4$. Symboly $F_1 \ldots F_4$ konjunkciou svojich literálov reprezentujú možné priradenia hodnôt premennej $\mathcal{F}$ podľa

$$\begin{aligned}
\mathcal{F} = 1 \quad &: \quad \neg F_1 \\
\mathcal{F} = 2 \quad &: \quad F_1 \wedge \neg F_2 \\
\mathcal{F} = 3 \quad &: \quad F_2 \wedge \neg F_3 \\
\mathcal{F} = 4 \quad &: \quad F_3 \wedge \neg F_4 \\
\mathcal{F} = 5 \quad &: \quad F_4
\end{aligned}$$

Ostatné symboly analogickým spôsobom reprezentujú priradenia hodnôt zvyšným premenným.

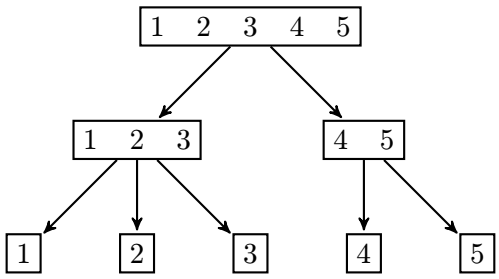
Definíciu symbolov je potrebné doplniť piatimi dodatočnými ohraničeniami podľa

$$\forall X, X = F, O, S, R, J \quad : \quad (X_1 \vee \neg X_2) \wedge (X_2 \vee \neg X_3) \wedge (X_3 \vee \neg X_4)$$

Rozšírenie: Hierarchické kódovanie premenných

Hierarchické kódovanie

Pri hierarchickom kódovaní sa doména premennej rozdelí na vzájomne disjunktné poddomény a až v nich sa uvažujú jednotlivé hodnoty. Jedno možné delenie pre premennú $\mathcal{V}$ s doménou $\{1, 2, 3, 4, 5\}$ je na obrázku



Pre zakódovanie nejakej hodnoty je potrebné zakódovať ako poddoménu tak aj hodnotu v danej poddoméne. Pritom obe úrovne (poddomény a hodnoty) je možné kódovať priamo alebo logaritmicky. Výhoda hierarchického kódovania je, že pre kódovanie na nižších úrovniach je možná recyklácia symbolov – kódovanie hodnôt 1, 2 a 3 v rámci poddomény {1, 2, 3} môže používať rovnaké symboly ako kódovanie hodnôt 4 a 5 v rámci poddomény {4, 5}.

Ak pre príklad na obrázku pre úroveň poddomén sa použije logaritmické (symbol Y_1) a pre úroveň hodnôt priame kódovanie (symboly Z_1, Z_2 a Z_3), tak napríklad priradenie $\mathcal{V} = 1$ sa vyjadrí ako $Y_1 \wedge Z_1$ a priradenie $\mathcal{V} = 5$ sa vyjadrí ako $\neg Y_1 \wedge Z_2$. K tomu je potrebné pripojiť $=_1 (Z_1, Z_2, Z_3)$ a $\neg(\neg Y_1 \wedge Z_3)$ pre zabezpečenie, že premenná nadobúda presne jednu hodnotu.

Rôznou kombináciou typov kódovaní je možné dosiahnuť potrebu 5 symbolov (obe úrovne kódované priamo), 4 symbolov (kombinácia priameho a logaritmického kódovania) alebo 3 symbolov (obe úrovne kódované logaritmicky). Celkový počet potrebných symbolov je vždy zdola ohraničený počtom nutným pri nehierarchickom logaritmickom kódovaní a zhora je ohraničený počtom potrebným pri nehierarchickom priamom kódovaní. Ak pre obe úrovne sa použije priame kódovanie, počet klauzúl pre zabezpečenie jednoznačnosti priradenia je typicky nižší ako pri nehierarchickom priamom kódovaní.

Delenie na poddomény môže byť aj viacúrovňové. Počet úrovní poddomén a spôsob a počet delení na nejakej úrovni sú voliteľné. V prípade, že nie každá kombinácia symbolov z rôznych úrovní je prípustná, tak neprípustné kombinácie je potrebné vylúčiť.

umožňujúcimi definovanie priradenia hodnoty premennej iba jedným alebo dvomi symbolmi.

- •

KÓDOVANIE PODMIENOK

V prvom kroku algoritmu Alg. 3.1 sú navrhované podmienky, ktoré reflektujú závislosti existujúce medzi navrhnutými premennými. Tieto úlohovo špecifické podmienky je potrebné kódovaním pretransformovať do tvaru logických výrazov definovaných nad logickými symbolmi. Keďže podmienky sa týkajú premenných, tak kódovanie podmienok musí zohľadniť použité kódovanie týchto premenných. Pri rôznych kódovaniach premenných bude viesť kódovanie podmienok na vety rôzneho tvaru.

V prípade premenných s dvojprvkovými doménami (vyjadrujúcimi či situácia reprezentovaná danou premennou platí alebo nie) je kódovanie pomerne jednoduchá záležitosť. Keďže v tomto prípade sú premenné reprezentované iba jedným symbolom, tak vlastne stačí vo formálnej podobe navrhnutých ohraničení zameniť označenia premenných za označenia zodpovedajúcich symbolov.

Zložitejšia situácia nastáva v prípade, ak doména premennej obsahuje viac elementov ako dva. Kódovanie takýchto premenných nahrádza premenné množinou symbolov. Jednotlivé metódy kódovania sa navzájom líšia v dvoch aspektoch:

- počtom symbolov, ktoré sú potrebné pre kódovanie jednej premennej,
- vhodnosťou kódovania pre vyjadrovanie podmienok určitého špecifického typu.

Počet použitých symbolov pre jednu premennú kolíše v intervale, ktorého minimálna hodnota je daná logaritmickým kódovaním a maximálna hodnota zase priamym kódovaním premenných. Z druhého hľadiska napríklad kardinalitné ohraničenia sa najjednoduchšie kódujú pri priamom kódovaní premenných zatiaľ čo nerovnosti sú najjednoduchšie vyjadrené kódovaním sekvenčným čítačom.

Preto ešte pred samotným kódovaním premenných je vhodné sa zamyslieť, aké typy ohraničení nad premennými sú definované a ako voľba kódovania premenných ovplyvní zložitosť následného kódovania ohraničení nad týmito premennými.

Príklad 3.13 V príklade na strane 82 boli definované podmienky pre ilustračný príklad určenia výsledkov turnaja. Tieto podmienky boli definované nad piatimi premennými $\mathcal{F}$, $\mathcal{O}$, $\mathcal{S}$, $\mathcal{R}$ a $\mathcal{J}$, ktoré majú rovnaké päťprvkové domény $\{1, 2, 3, 4, 5\}$. Tieto podmienky sú troch typov: "alldiff", "priradenie hodnoty" a "nerovnosť". Ukážme ako si s týmito typmi podmienok poradia spomínané metódy kódovania premenných.

Podmienka *alldiff*($\mathcal{F}$, $\mathcal{O}$, $\mathcal{S}$, $\mathcal{R}$, $\mathcal{J}$) požaduje, aby premenné nadobudli navzájom rôzne hodnoty a teda aby nedošlo k zdieľaniu žiadnej hodnoty dvomi premennými. Použilo sa iba ohraničenie $\leq_1$. Komplementárne ohraničenie $\geq_1$ nie je potrebné reprezentovať, lebo je redundantné vzhľadom na to, že každá premenná nadobúda práve jednu hodnotu (čo je zabezpečené zvolenými reprezentáciami).

- priame kódovanie:
  Jedinečnosť prvej hodnoty sa vyjadrí $\leq_1 (F_1, J_1, O_1, R_1, S_1)$. Vedie na klauzuly s dvomi literálmi.

- logaritmické kódovanie:
  Jedinečnosť druhej hodnoty sa vyjadrí $\leq_1 (\neg F_1 \wedge \neg F_2 \wedge F_3, \neg O_1 \wedge \neg O_2 \wedge O_3, \neg S_1 \wedge \neg S_2 \wedge S_3, \neg R_1 \wedge \neg R_2 \wedge R_3, \neg J_1 \wedge \neg J_2 \wedge J_3)$. Vedie na klauzuly, ktoré budú mať 6 literálov (pre hodnoty 1 a 2) alebo 4 literály (pre hodnoty 3, 4 a 5).

- sekvenčné kódovanie:
  Jedinečnosť tretej hodnoty sa vyjadrí $\leq_1 (F_2 \wedge \neg F_3, O_2 \wedge \neg O_3, S_2 \wedge \neg S_3, R_2 \wedge \neg R_3, J_2 \wedge \neg J_3)$. Vedie na klauzuly, ktoré budú mať buď 4 literály (pre hodnoty 2,3 a 4) alebo 2 literály (pre hodnoty 1 a 5).

Vo všetkým prípadoch výsledkom[3] bude 50 klauzúl (po 10 pre každú z možných hodnôt), jednotlivé metódy kódovania sa budú navzájom líšiť dĺžkou vytváraných klauzúl.

Podmienka $(\mathcal{J} = 1) \oplus (\mathcal{F} = 2)$ požaduje, aby premenné nadobúdali konkrétne hodnoty zo svojich domén.

- priame kódovanie: $J_1 \oplus F_2$
  Vedie na 2 klauzuly, každá s 2 literálmi.

- logaritmické kódovanie: $(\neg J_1 \wedge \neg J_2 \wedge \neg J_3) \oplus (\neg F_1 \wedge \neg F_2 \wedge F_3)$
  Vedie na 10 klauzúl – 9 klauzúl má 2 literály a 1 klauzula má 6 literálov.

- sekvenčné kódovanie: $\neg J_1 \oplus (F_1 \wedge \neg F_2)$
  Vedie na 3 klauzuly – 2 klauzuly majú 2 literály a 1 klauzula má 3 literály.

V týchto prípadoch sa jednotlivé metódy kódovania premenných líšia nielen dĺžkou klauzúl ale aj ich počtom.

Podmienka $\mathcal{J} < \mathcal{O}$ reprezentuje vzťah nerovnosti medzi validnými dvojicami hodnôt dvoch premenných.

[3] Uvažuje sa binomická reprezentácia kardinalitných ohraničení.

- priame kódovanie:

$$\begin{aligned} O_2 &\rightarrow J_1 \\ O_3 &\rightarrow J_1 \vee J_2 \\ O_4 &\rightarrow J_1 \vee J_2 \vee J_3 \\ O_5 &\rightarrow J_1 \vee J_2 \vee J_3 \vee J_4 \end{aligned}$$

  Vedie na 4 klauzuly dĺžky 2, 3, 4 a 5 literálov.

- logaritmické kódovanie:

$$\begin{aligned} \neg O_1 \wedge \neg O_2 \wedge O_3 &\rightarrow \neg(\neg J_1 \wedge \neg J_2 \wedge J_3) \wedge \\ &\quad \neg(\neg J_1 \wedge J_2) \wedge \neg(J_1 \wedge \neg J_2) \wedge \neg(J_1 \wedge J_2) \\ \neg O_1 \wedge O_2 &\rightarrow \neg(\neg J_1 \wedge J_2) \wedge \neg(J_1 \wedge \neg J_2) \wedge \neg(J_1 \wedge J_2) \\ O_1 \wedge \neg O_2 &\rightarrow \neg(J_1 \wedge \neg J_2) \wedge \neg(J_1 \wedge J_2) \\ O_1 \wedge O_2 &\rightarrow \neg(J_1 \wedge J_2) \end{aligned}$$

  Vedie na 6 klauzúl so štyrmi literálmi, 3 klauzuly s 5 literálmi a 1 klauzulu s 6 literálmi.

- sekvenčné kódovanie:

$$\begin{aligned} O_1 \wedge \neg O_2 &\rightarrow \neg J_1 \\ O_2 \wedge \neg O_3 &\rightarrow \neg J_2 \\ O_3 \wedge \neg O_4 &\rightarrow \neg J_3 \\ O_4 &\rightarrow \neg J_4 \end{aligned}$$

  Vedie na 3 klauzuly s 3 literálmi a 1 klazulu s 2 literálmi.

V prvých dvoch metódach dĺžky klauzúl postupne narastajú, v poslednej metóde dĺžky ostávajú krátke. •

PSEUDO BOOLEOVE OHRANIČENIA

Ak by sme uvažovali reprezentáciu pravdivostných hodnôt pomocou hodnôt 1 (namiesto TRUE) a 0 (namiesto FALSE), tak kardinalitné ohraničenia by sa dali vyjadriť v aritmetickej podobe. Napríklad ohraničenie $=_k (X_1, \ldots, X_n)$ by bolo možné zapísať ako $X_1 + \ldots + X_n = k$. Toto je však iba špeciálny prípad všeobecnejšieho zápisu v tvare lineárnej kombinácie hodnôt symbolov

$$w_1 X_1 + \ldots + w_n X_n = k$$

kde všetky váhy w_i sú rovné jednej. Otázkou je, ako vyjadriť ohraničenie, ktoré má tvar lineárnej váženej kombinácie hodnôt symbolov avšak s rôznymi váhami.

Príklad 3.14 Príkladom takéhoto lineárne váženého ohraničenia je

$$4X_1 + 2X_2 + 3X_3 + 7X_4 + X_5 \leq 5$$

Toto ohraničenie je možné analyzovať a hľadať také kombinácie symbolov, ktorých súčasná interpretácia ako pravdivých by spôsobila nesplnenie daného ohraničenia. Pri uvažovaní samotných symbolov osobitne dochádza iba v jednom prípade k nesplneniu ohraničenia – keď je pravdivé X_4. Pri uvažovaní dvojíc to sú dvojice $X_1 \wedge X_2$ a $X_1 \wedge X_3$. Patria sem aj všetky dvojice v ktorých vystupuje X_4, ale tie už nie je potrebné uvažovať (pretože napríklad $X_4 \wedge X_5$ je pohltené skôr nájdeným X_4). V rámci trojíc ešte pribudne $X_2 \wedge X_3 \wedge X_5$. Ostatné trojice, štvorice a aj päticu symbolov kvôli pohlteniu už nájdenými kombináciami nie je potrebné uvažovať.

Ohraničenie je nesplnené, keď aspoň jedna z identifikovaných kombinácií symbolov je pravdivá. Pre splnenie ohraničenia nesmie dôjsť k tomuto a preto musí platiť

$$\neg\left(X_4 \vee (X_1 \wedge X_2) \vee (X_1 \wedge X_3) \vee (X_2 \wedge X_3 \wedge X_5)\right)$$

čo sa dá transformovať na klauzulárny tvar

$$\neg X_4 \wedge (\neg X_1 \vee \neg X_2) \wedge (\neg X_1 \vee \neg X_3) \wedge (\neg X_2 \vee \neg X_3 \vee \neg X_5)$$

•

Cvičenia

1. Pre úlohu naplánovania turnaja z ilustračného príkladu na strane 68 navrhnite podmienky, ktoré musí spĺňať reprezentácia pomocou 100 premenných typu $\mathcal{V}_{k,s1,s2}$ ($s_1 \neq s_2$), kde napríklad premenná $\mathcal{V}_{3,O,S}$ hovorí, že v treťom kole hrali spolu Ondro a Stano. Navrhnuté podmienky doplňte tak, aby sa nevyskytovala symetria riešení v rámci kola.

2. Pre úlohu určenia poradia z ilustračného príkladu na strane 68 navrhnite podmienky, ktoré musí spĺňať reprezentácia pomocou 20 premenných typu $\mathcal{V}_{s1,s2}$ ($s_1 \neq s_2$), kde napríklad premenná $\mathcal{V}_{O,S}$ hovorí, že

- Ondro sa umiestnil o jednu pozíciu pred Stanom resp. že Ondro sa umiestnil o jednu pozíciu za Stanom,
- Ondro sa umiestnil na nejakej pozícii pred Stanom resp. že Ondro sa umiestnil na nejakej pozícii za Stanom.

3. Predpokladajme, že v úlohe určenia poradia z ilustračného príkladu na strane 68 sa pracuje s umiestnením v aktuálnom a aj minulom turnaji. Skúste vyjadriť tvrdenia

   a. Ondro aktuálne skončil o dve miesta lepšie ako Jaro,
   b. Fero bol aktuálne lepší ako štvrtý,
   c. Stano skončil na rovnakom mieste ako minule,
   d. Rudo si polepšil oproti minule,
   e. Fero sa zlepšil viac ako Stano,
   f. Jaro sa posunul o rovnaký počet miest ako Rudo.

   Na základe toho určte vhodnosť použitia rôznej voľby premenných pre dané výroky, pričom uvažujte

   a. $\mathcal{V}_{s,m,t}$ kde s je študent, m je jeho umiestnenie a t označuje turnaj (aktuálny alebo minulý),
   b. $\mathcal{V}_{s1,s2,t}$ kde s_1 aj s_2 sú rôzni študenti (s_1 sa umiestnil lepšie ako s_2) a t označuje turnaj,
   c. $\mathcal{V}_{s1,s2,t}$ kde s_1 aj s_2 sú rôzni študenti (s_1 sa umiestnil tesne pred s_2) a t označuje turnaj.

4. Ak v úlohe určenia poradia z ilustračného príkladu na strane 68 sa pracuje s relatívnym poradím reprezentovaným premennými typu $\mathcal{V}_{s1,s2}$ ($s_1 \neq s_2$) s doménou $\{\top, \bot\}$, tak je potrebné previesť toto relatívne poradie na poradie absolútne, reprezentované 25 premennými typu $\mathcal{V}_{s,m}$ s doménou $\{\top, \bot\}$ (s – student, m – poradie). Navrhnite podmienky tohto prevodu, ak premenná $\mathcal{V}_{s1,s2}$ reprezentuje fakt, že

   a. študent s_1 sa umiestnil lepšie ako študent s_2 (hodnota $\top$) alebo študent s_1 sa umiestnil horšie ako študent s_2 (hodnota $\bot$),
   b. študent s_1 sa umiestnil o jednu pozíciu pred študentom s_2 (hodnota $\top$) alebo študent s_1 sa umiestnil o jednu pozíciu za študentom s_2 (hodnota $\bot$).

5. Hlavolam Sudoku pozostáva z mriežky vytvorenej deviatimi riadkami, deviatimi stĺpcami a deviatimi 3x3 štvorcami. Spolu obsahuje 81 (9x9) pozícií, pričom niektoré pozície sú obsadené, zatiaľ čo iné sú voľné. Cieľom je na každú voľnú pozíciu priradiť jednu číslicu z $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ tak, aby v žiadnom riadku, stĺpci ani štvorci neboli dve rovnaké číslice.

   Uvažujte prázdny hlavolam, keď na žiadnej pozícii nie je predpísaná hodnota. Vytvorte všetky podmienky, ktoré sa viažu na použitie premenných typu $\mathcal{V}_{i,j,h}$ stanovujúcich, či na pozícii v i-tom riadku a j-tom stĺpci sa nachádza číslica h (hodnota $\top$) alebo nie (hodnota $\bot$). Určte minimálne množiny podmienok a preskúmajte vplyv rozširovania množiny podmienok nad minimálnu množinu.

6. Rozvrhnite m figúr na šachovnici veľkosti kxk políčok bez toho, aby sa navzájom ohrozovali (problém sa dá vyjadriť pomocou tvrdení "aspoň jedna figúra sa nachádza na X" a "najviac jedna figúra sa nachádza na X"), pričom ako figúry použite

   - veže – pod X sa chápe konkrétny riadok alebo stĺpec,
   - strelci – pod X sa chápe konkrétna diagonála,
   - dámy – pod X sa chápe konkrétny riadok, stĺpec alebo diagonála,
   - jazdci – pod X sa chápu konkrétne políčka.

   Úlohu riešte pre prípad $k = 4$, pričom je potrebné nájsť všetky riešenia pomocou SAT solvera. Použite varianty $m < k$, $m = k$ a $m > k$. Maximálne koľko figúr určitého typu sa dá takto rozmiestniť na šachovnici ?

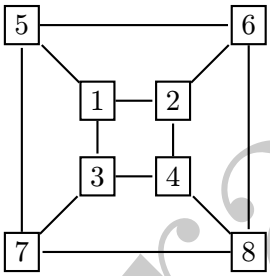
7. Pripravte rozvrh pre turnaj v golfe, ktorého sa zúčastňuje n ($n = g*p$) hráčov, pričom hráči hrajú v g skupinách po p hráčoch. Turnaj trvá w kôl. Podmienkou je, že každá možná dvojica hráčov hrá spolu v jednej skupine najviac jeden krát. Riešte pre usporiadanie hráčov do $g = 5$ skupín po $p = 3$ hráčoch pre rôzne hodnoty w. Použite symboly s významom

   - $X_{i,k,l}$ - hráč i hrá v skupine k v kole l,
   - $X_{i,j,k,l}$ - hráč i hrá na pozícii j v skupine k v kole l,
   - $X_{i_1,i_2,l}$ - hráč i_1 hrá spolu s hráčom i_2 v kole l,
   - $X_{i_1,i_2,k,l}$ - hráč i_1 hrá spolu s hráčom i_2 v skupine k v kole l,
   - $X_{i_1,i_2,i_3,l}$ - hráč i_1 hrá spolu s hráčom i_2 a hráčom i_3 v kole l,

- $X_{i_1,i_2,i_3,k,l}$ - hráč i_1 hrá spolu s hráčom i_2 a hráčom i_3 v skupine k v kole l.

Preskúmajte vplyv pridania podmienok pre vylúčenie (resp. redukciu) symetrie.

8. V grafe na obrázku nájdite Hamiltonovskú cyklickú cestu, ktorá začne v nejakom uzle, prejde všetkými ostatnými uzlami a vráti sa do počiatočného uzla, pričom do každého uzla vchádza raz a raz z neho vychádza (pretože cesta je cyklická, je jedno v ktorom uzle začne).



9. Aké je chromatické číslo grafu (najmenší možný počet farieb nutných na vyfarbenie uzlov grafu tak, aby každé dva uzly, spojené hranou, mali navzájom rôznu farbu), pričom graf

   a. je na obrázku v predchádzajúcom prípade,
   b. k obrázku v predchádzajúcom prípade pridáva dve hrany – hranu medzi uzlami 1 a 4 a hranu medzi uzlami 2 a 3.

10. Pre úlohu naplánovania cesty výťahom z ilustračného príkladu na strane 68 navrhnite podmienky, ktoré musí spĺňať reprezentácia pomocou 30 premenných tvaru $\mathcal{V}_{o,j}$ a 25 premenných tvaru $\mathcal{W}_{b,s}$, kde napríklad premenná $\mathcal{V}_{O,3}$ reprezentuje Ondrovu účasť v druhej jazde výťahu, premenná $\mathcal{V}_{batO,2}$ reprezentuje účasť Ondrovej batožiny v tretej jazde výťahu a premenná $\mathcal{W}_{batF,O}$ vyjadruje, že Ferovu batožinu prepravuje Ondro.

11. Je daná možina balíkov rôznej veľkosti a sada rovnakých kontajnerov. Balíky je potrebné uložiť do kontajnerov tak, aby dostupný počet kontajnerov postačoval na zabalenie všetkých balíkov. Pôdorys všetkých balíkov aj vnútorného priestoru kontajnerov je rovnaký. Výška jednotlivých balíkov očíslovaných 1 až 6 je po poradí 8, 6, 9, 5, 5 a 9 jednotiek. Výška vnútorného priestoru kontajnerov je 16 jednotiek, k dispozícii sú tri kontajnery.

12. Formálne verifikujte funkčnosť obvodu sekvenčného čítača zostaveného zo štyroch rovnakých hradiel, ktoré sú spojené podľa nasledovného obrázku

pričom uvažujte iba tie vývody, ktoré sú označené ($R*$ sú vnútorné signály obvodu, $X*$ reprezentujú vstupné signály obvodu, O je výstupným signálom, $\perp$ je signál reprezentujúci logickú nulu).

Z činnosti obvodu odvoďte pravidlá pre kódovanie kardinalitných ohraničení $\leq_k$ a $\geq_k$.

Literatúra

1. Ben-Haim, Y. et al.: Perfect Hashing and CNF Encodings of Cardinality Constraints. In: Theory and Applications of Satisfiability Testing – SAT 2012, Springer, LNCS 7317, 2012, 397–409.

2. Björk, M.: Successful SAT Encoding Techniques. Journal on Satisfiability, Boolean Modeling, and Computation, vol. 7, 2009, 189–201.

3. Frisch, A.M. – Giannaros, P.A.: SAT Encodings of the At-Most-k Constraint: Some Old, Some New, Some Fast, Some Slow. In: Proc. of the 10th Int. Workshop on Constraint Modelling and Reformulation, ModRef, 2010.

4. Gavanelli, M.: The Log-Support Encoding of CSP into SAT. In: Principles and Practice of Constraint Programming. Springet, LNCS 4741, 2007, 815–822.

5. Hölldobler, S. – Nguyen, V.H.: An Efficient Encoding of the at-most-one Constraint. KRR Report 13-04, Institute of Artificial Intelligence, Technische Universität Dresden, 2004.

6. Jabbour, S. – Sais, L. – Salhi, Y.: Pigeon-Hole Based Encoding of Cardinality Constraints. In: Proc. of the Theory and Practice of Logic Programming, online supplement 13, 2013, 1–10.

7. Klieber, W. – Kwon, G.: Efficient CNF Encoding for Selecting 1 from N Objects. In: Proc. of the 4th Int. Workshop on Constraints in Formal Verification, 2007.

8. Petke, J.: Bridging Constraint Satisfaction and Boolean Satisfiability. Springer, Cham, 2015.

9. Roussel, O. – Manquinho, V.: Pseudo-Boolean and Cardinality Constraints. In: Handbook of Satisfiability, Biere et al. (eds.). IOS Press, Amsterdam, 2009, 695–733.

10. Sakallah, K.A.: Symmetry and Satisfiability. In: Handbook of Satisfiability, Biere et al. (eds.). IOS Press, Amsterdam, 2009, 289–337.

11. Triska, M. – Musliu, N.: An Improved SAT Formulation for the Social Golfer Problem. Annals of Operations Research, roč. 194, 2012, č. 1, 427–438.

12. Van-Hau, N. – Son, T.M.: A New Method to Encode the At-Most-One Constraint into SAT. In: Proc. of the 6th Int. Symposium on Information and Communication Technology, Hue City, Vietnam, 2015, 46–53.

13. Velev, M.N. – Gao, P.: Modular schemes for constructing equivalent Boolean encodings of cardinality constraints and application to error diagnosis in formal verification of pipelined microprocessors. In: Proc. of the 9th Symposium on Abstraction, Reformulation, and Approximation, Parador de Cardona, Spain, 2011, 125–131.