# Programiranje ograničenja – specijalni kurs –

Milan Banković

*Matematički fakultet,
Univerzitet u Beogradu

Prolećni semestar 2022/23.

# Pregled

# O programiranju ograničenja

## Šta je programiranje ograničenja?

Programiranje ograničenja je deklarativna programska paradigma koja omogućava rešavanje široke klase problema iz različitih domena.

## Kakve probleme razmatramo?

Iako je gornja definicija prilično opšta, najčešće se radi o različitim problemima kombinatorne pretrage i kombinatorne optimizacije. U širem smislu, u domen programiranja ograničenja spadaju i sledeći problemi:

- rešavanje jednačina i sistema jednačina
- linearno i celobrojno programiranje
- SAT problem (iskazna zadovoljivost) i SMT problem (zadovoljivost u teoriji prvog reda)
- ...

# Modelovanje problema

## Na koji način se predstavlja problem koji se rešava?

Problem koji rešavamo predstavljamo skupom promenljivih koje uzimaju vrednosti iz zadatih domena:

- rešenje problema se predstavlja vrednostima koje se dodeljuju promenljivama
- uslovi koje rešenje mora da zadovoljava zadaju se ograničenjima (relacijama nad domenima promenljivih)

# Rešavanje problema

## Kako se rešavaju ovakvi problemi?

U zavisnosti od vrste domena i ograničenja koja se u problemu javljaju, za rešavanje se koriste različite tehnike zasnovane na veštačkoj inteligenciji (rezonovanju, pretrazi), ali i druge matematičke tehnike.
Razlikujemo:

- metode specifične za domen – pogodne za rešavanje specifičnih problema
- opšte metode – pogodne za široku klasu problema

# Pregled

# Problem zadovoljenja ograničenja

## Definicija 1

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

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

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

## Napomena

- Rešenje $(d_1, \ldots, d_n)$ ćemo često zapisivati i kao dodelu $\{x_1 = d_1, \ldots, x_n = d_n\}$
- ,,Rešiti CSP problem" obično podrazumeva pronalaženje jednog rešenja
  - ređe, može se zahtevati pronalaženje svih rešenja

# Predstavljanje ograničenja

## Kako predstavljamo ograničenja?

- ograničenja možemo predstavljati skupom $r$-torki koje ga zadovoljavaju (tzv. tabelarna ograničenja)
- češće, ograničenja predstavljamo simbolički, kao relacije nad promenljivama (npr. $x < y$)
- ograničenja mogu biti unarna (npr. $x > 0$, $x \neq 3$, ...), binarna (npr. $x < y$, $x = y$, $x \mid y$, ...), kao i arnosti veće od dva (npr. $x + y = z$)
- vrlo česta su različita aritmetička ograničenja (npr. $x^2 + y^2 \leq 9$)
- pored jednostavnih relacija, često koristimo i ograničenja sa složenom semantikom koja nazivamo globalna ograničenja (više o njima kasnije)

# Domeni promenljivih

## Kakvi mogu biti domeni promenljivih?

Tehnike rešavanja CSP problema umnogome zavise od prirode domena promenljivih koje se razmatraju:

- konačni domeni: problemi kombinatorne pretrage
  - dobro razvijene opšte metode rešavanja, zasnovane na rezonovanju i pretrazi
- beskonačni domeni – diskretni i kontinualni
  - uglavnom se koriste specifične tehnike, u zavisnosti od tipa ograničenja koja se javljaju

## Napomena

Mi ćemo se u nastavku baviti isključivo problemima nad konačnim domenima i opštim tehnikama za njihovo rešavanje.

# Primer – Pitagorine trojke

## Primer 1

*Razmotrimo CSP problem $\mathcal{P}$ nad promenljivama $\mathcal{X} = (x, y, z)$, gde je $D(x) = D(y) = D(z) = \mathbb{N}$ (skup prirodnih brojeva), a skup ograničenja je zadat sa:*

- $z^2 = x^2 + y^2$
- $x < y$

*Ovaj problem je nad beskonačnim domenima. Rešenja ovog problema su sve Pitagorine trojke prirodnih brojeva i ima ih beskonačno mnogo. Uslov $x < y$ sprečava da pored rešenja $\{x = 3, y = 4, z = 5\}$ imamo i rešenje $\{x = 4, y = 3, z = 5\}$, što su suštinski iste Pitagorine trojke. Ovakvi uslovi se koriste za razbijanje simetrija (engl. symmetry breaking).*

# Primer – kriptoaritmetika

## Primer 2

*Razmotrimo sledeći problem: slova treba zameniti dekadnim ciframa (ista slova istim ciframa, a različita različitim) tako da sabiranje bude ispravno:*

$$\begin{array}{rr} & SEND \\ + & MORE \\ \hline & MONEY \end{array}$$

*Ovakvi problemi poznati su i kao kriptoaritmetičke slagalice (engl. cryptoarithmetic puzzles). Problem se može jednostavno predstaviti kao CSP problem na sledeći način:*

- *Skup promenljivih* $\mathcal{X} = (S, E, N, D, M, O, R, Y)$
- *Domeni svih promenljivih je skup* $\{0, 1, 2, \ldots, 9\}$*, izuzev promenljivih* $M$ *i* $S$*, čiji je domen* $\{1, 2, \ldots, 9\}$ *(primetimo da su svi domeni konačni skupovi)*
- *Skup ograničenja se sastoji iz sledećih ograničenja:*
  - $S \neq E$, $S \neq N$, $\ldots$, $S \neq Y$, $E \neq N$, $\ldots$, $R \neq Y$
  - $$\begin{aligned} & 1000 \cdot S + 100 \cdot E + 10 \cdot N + D \\ & 1000 \cdot M + 100 \cdot O + 10 \cdot R + E \\ = \; & 10000 \cdot M + 1000 \cdot O + 100 \cdot N + 10 \cdot E + Y \end{aligned}$$

# Primer – linearni sistem

## Primer 3

*Razmotrimo CSP problem nad promenljivama $\mathcal{X} = (x, y)$, gde je $D(x) = D(y) = \mathbb{R}$ (skup realnih brojeva), a ograničenja su zadata na sledeći način:*

- $2x + 3y = 5$
- $x - y = 10$

*Ovaj problem je nad kontinualnim domenima, a prostor rešenja je neprebrojiv. Ipak, ovaj problem ima tačno jedno rešenje:* $\{x = 7, y = -3\}$.

## Napomena

Ovaj primer nam pokazuje da se i poznati problem rešavanja sistema linearnih jednačina nad poljem realnih brojeva može razumeti kao specijalni slučaj CSP problema.

# Problem zadovoljenja ograničenja

## Definicija 2

*Za CSP problem $\mathcal{P}$ kažemo da je konzistentan ako ima bar jedno rešenje. U suprotnom je nekonzistentan.*

## Kakav može biti skup rešenja?

- Konzistentan CSP može imati jedno ili više rešenja
- Konzistentan CSP nad konačnim domenima uvek ima konačno mnogo rešenja

# Primer – nekonzistentan problem

## Primer 4

*Neka je problem $\mathcal{P}$ nad skupom promenljivih $x, y, z$, takvih da je $D(x) = D(y) = D(z) = \{1, 2\}$. Ako imamo skup ograničenja $x \neq y$, $x \neq z$, $y \neq z$, tada je jasno da je problem $\mathcal{P}$ nekonzistentan.*

# Pregled

# Modelovanje problema

## Šta je CSP modelovanje?

Pod CSP modelovanjem podrazumevamo formulaciju datog problema iz prakse u obliku CSP problema. Dobijeni CSP problem nazivamo modelom datog problema iz prakse.

## Izazovi prilikom modelovanja

- Prilikom modelovanja je ključno prepoznati promenljive i njihove domene, kao i izraziti odgovarajuća ograničenja
- Modelovanje nije trivijalan posao:
  - mnogi problemi se mogu modelovati na više načina
  - neki modeli omogućavaju jednostavniju formulaciju ograničenja, a neki efikasnije rešavanje (ne obavezno isti modeli!)

# Ograničenja poređenja

## Definicija 3

*Ograničenja oblika* $x \bowtie c$ *ili* $x \bowtie y$*, gde su* $x$ *i* $y$ *promenljive,* $c$ *konstanta, a* $\bowtie \in \{=, \neq, <, >, \leq, \geq\}$ *nazivamo ograničenja poređenja.*

## Napomena

Ograničenja poređenja su najjednostavnija ograničenja koja se pojavljuju u CSP problemima i veoma su često korišćena.

# Primer – takmičenje

## Primer 5

*Petao, Gavran i Kukavica se takmiče u pevanju. U žiriju se nalazi određeni broj sudija. Svaki sudija je glasao za jednog od takmičara. Detlić je bio zadužen za brojanje glasova. Ukupno je izbrojao 59 glasova. Po njegovom brojanju, 15 ih je glasalo za Petla ili Gavrana, 18 za Gavrana ili Kukavicu, a 20 za Kukavicu ili Petla. Detliću brojanje nije jača strana, tako da ni jedan od ova četiri broja ne mora biti tačan, ali se zna da ni u jednom slučaju nije pogrešio za više od 13. Koji su rezultati glasanja?*

# Primer – takmičenje (2)

## Primer (nastavak)

*(nastavak) Uvedimo promenljive $P$, $G$ i $K$ koje označavaju redom broj glasova za Petla, Gavrana i Kukavicu. Uzmimo da je domen ovih promenljivih $\{1, \ldots, 72\}$ (s obzirom da zbir sve tri promenljive sigurno nije veći od $59 + 13 = 72$). Sada imamo sledeća ograničenja:*

- $46 \leq P + G + K \leq 72$
- $2 \leq P + G \leq 28$
- $5 \leq G + K \leq 31$
- $7 \leq K + P \leq 33$

# Linearno ograničenje

## Definicija 4

*Ograničenje oblika:*

$$a_1x_1 + \ldots + a_nx_n \bowtie c$$

*gde su* $a_1, \ldots, a_n$ *i* $c$ *konstante,* $x_1, \ldots, x_n$ *promenljive, a* $\bowtie \in \{=, \neq, <, >, \leq, \geq\}$ *nazivamo linearno ograničenje (ograničenje sume, ograničenje skalarnog proizvoda).*

## Napomena

Ovo ograničenje je jedno od najčešćih ograničenja koja se javljaju u modelima.

## Napomena

Često se javlja i ograničenje oblika:

$$y = a_1x_1 + \ldots + a_nx_n$$

koje je specijalni slučaj linearnog ograničenja (jer je ekvivalentno sa $y - a_1x_1 - \ldots - a_nx_n = 0$).

# Primer – magični kvadrat

## Primer 6

*Dat je magični kvadrat:*

| | | | |
|---|---|---|---|
| | 12 | | |
| | 8 | 15 | |
| 7 | | 2 | |
| 4 | | | 11 |

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

# Primer – magični kvadrat (2)

## Primer (nastavak)

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

- *Uvodimo promenljive* $x_{ij}$ *za polje u* $i$*-toj vrsti i* $j$*-toj koloni* $(1 \leq i, j \leq 4)$
- *Sve promenljive imaju domen* $\{1, 2, \ldots, 16\}$
- *Skup ograničenja:*
  - $x_{i1} + \ldots + x_{i4} = 34$, *za svako* $i \in \{1, 2, 3, 4\}$
  - $x_{1j} + \ldots + x_{4j} = 34$, *za svako* $j \in \{1, 2, 3, 4\}$
  - $x_{11} + x_{22} + x_{33} + x_{44} = 34$
  - $x_{41} + x_{32} + x_{23} + x_{14} = 34$
  - $x_{12} = 12$, $x_{22} = 8$, $x_{23} = 15$, $x_{31} = 7$, $x_{33} = 2$, $x_{41} = 4$, $x_{44} = 11$
  - $\texttt{alldiff}(x_{11}, \ldots, x_{14}, x_{21}, \ldots, x_{24}, x_{31}, \ldots, x_{34}, x_{41}, \ldots, x_{44})$

## Definicija 5

*Ograničenje* $\texttt{alldiff}(x_1, \ldots, x_k)$ *definišemo na sledeći način:*

$$\texttt{alldiff}(x_1, \ldots, x_k) = \{(d_1, \ldots, d_k) \mid d_i \in D(x_i), i \neq j \Rightarrow d_i \neq d_j\}$$

*Intuitivno, ovo ograničenje zahteva da sve navedene promenljive uzimaju međusobno različite vrednosti.*

# Primer – latinski kvadrat

## Primer 7

*Razmotrimo primer latinskog kvadrata:*

| | | | |
|---|---|---|---|
| 1 | | | |
| | | | 3 |
| | 4 | | |
| | | 2 | |

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

# Primer – latinski kvadrat (2)

## Primer (nastavak)

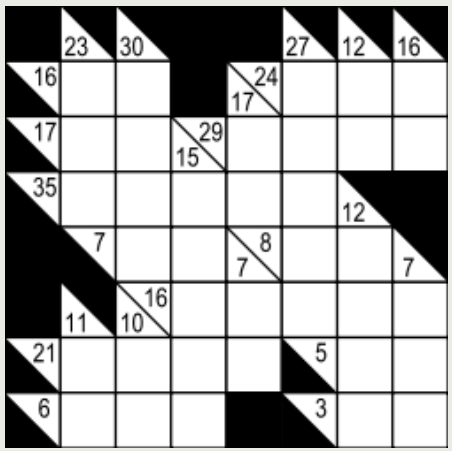
*(nastavak) Latinski kvadrat se svodi na sledeći CSP problem:*

- *skup promenljivih:* $x_{ij}$ *za polje u* $i$*-toj vrsti i* $j$*-toj koloni*
- *domeni svih promenljivih su* $\{1, 2, 3, 4\}$
  - $\texttt{alldiff}(x_{i1}, \ldots, x_{i4})$, *za* $i \in \{1, 2, 3, 4\}$
  - $\texttt{alldiff}(x_{1j}, \ldots, x_{4j})$, *za* $j \in \{1, 2, 3, 4\}$

# Primer – kakuro

## Primer 8

*Kakuro predstavlja poznatu japansku slagalicu, nalik ukrštenici, samo sa brojevima:*




*Bela polja treba popuniti brojevima od* 1 *do* 9 *tako da se svaka ,,brojevna reč" po vrstama i kolonama sastoji iz različitih brojeva, a zbirovi brojeva u svakoj reči su jednaki datim brojevima.*

# Primer – kakuro (2)

## Primer (nastavak)

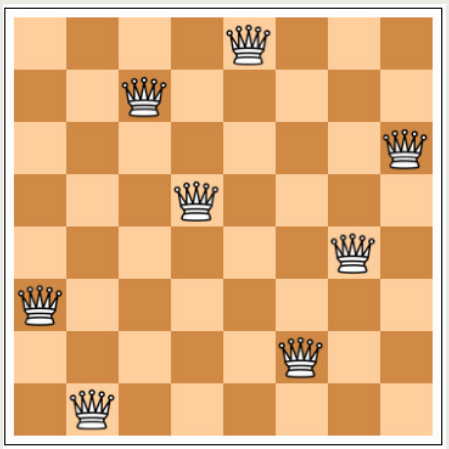
*(nastavak) Kakuro slagalica iz prethodnog primera se može predstaviti kao CSP problem na sledeći način:*

- *svakom belom polju se pridruži jedna promenljiva sa domenom* $\{1, 2, \ldots, 9\}$
  - *neka su promenljive pridružene belim poljima redom po vrstama, sa leva na desno* $x_1, x_2, \ldots, x_{36}$
- *za svaku ,,reč", promenljive koje odgovaraju poljima te reči su ograničene* `alldiff` *ograničenjem*
  - *na primer, za prvu vrstu gornje tablice imamo dva ograničenja:* $\texttt{alldiff}(x_1, x_2)$, $\texttt{alldiff}(x_3, x_4, x_5)$
- *za svaku ,,reč", zbir vrednosti promenljivih koje odgovaraju poljima te reči mora biti jednak datom zbiru*
  - *na primer, za prvu vrstu gornje tablice, imamo dva ograničenja:* $x_1 + x_2 = 16$, $x_3 + x_4 + x_5 = 24$

# Primer – 8 kraljica

## Primer 9

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



# Primer – 8 kraljica (2)

## Primer (nastavak)

*(nastavak) Problem 8 kraljica se može predstaviti na sledeći način:*

- *neka je $x_i$ (za $1 \leq 1 \leq 8$) promenljiva koja označava redni broj vrste u kome se nalazi i-ta kraljica ($D(x_i) = \{1, \ldots, 8\}$)*
- *neka je $y_j$ (za $1 \leq j \leq 8$) promenljiva koja označava redni broj kolone u kojoj se nalazi j-ta kraljica ($D(y_j) = \{1, \ldots, 8\}$)*
- *mora da važi: $\texttt{alldiff}(x_1, \ldots, x_8)$ i $\texttt{alldiff}(y_1, \ldots, y_8)$ (nikoje dve kraljice ne smeju biti u istoj vrsti ni u istoj koloni)*
- *za $i \neq j$ mora da važi: $|x_i - x_j| \neq |y_i - y_j|$ (nikoje dve kraljice ne smeju biti na istoj dijagonali)*

# Primer – 8 kraljica (3)

## Primer (nastavak)

*(nastavak) Prethodni model problema 8 kraljica se može pojednostaviti tehnikom razbijanja simetrija: s obzirom da znamo da nikoje dve kraljice ne mogu biti u istoj koloni, možemo, bez gubitka opštosti, pretpostaviti da je prva kraljica u prvoj koloni, druga u drugoj, i sl. Time se gubi potreba za y promenljivama, a ostaju nam samo x promenljive i ograničenja:*

- $\texttt{alldiff}(x_1, \ldots, x_8)$ – *nikoje dve kraljice se ne smeju nalaziti u istoj vrsti*
- *za* $i \neq j$ *mora da važi* $|x_i - x_j| \neq |i - j|$ *(primetimo da je* $|i - j|$ *konkretan broj za svako* $i, j$*)*

# Transformacija ograničenja

## Koliko slobode imamo u zadavanju ograničenja?

- CSP rešavači obično podržavaju ograničen skup tipova ograničenja
- Sva ograničenja koja se javljaju u modelu moraju se svesti na ona ograničenja koja rešavač podržava
- Ograničenje oblika $|x - y| \neq c$ obično nije među njima

## Napomena

- Mi ćemo u nastavku ograničenja iz naših modela uvek svoditi na standardna ograničenja koja su podržana od strane većine rešavača
- Ovakva ograničenja ćemo, kao i do sada, eksplicitno uvoditi definicijama

# Apsolutna vrednost

## Definicija 6

*Ograničenje oblika* $y = |x|$ *nazivamo ograničenjem apsolutne vrednosti.*

## Napomena

Ograničenje oblika $|x - y| \neq c$ se može svesti na dva ograničenja: $x - y \neq c$ i $x - y \neq -c$. Alternativno, možemo uvesti nove promenljive $z$ i $u$ i svesti polazno ograničenje na ograničenja: $z = x - y$, $u = |z|$ i $u \neq c$. Na sličan način se može transformisati i ograničenje oblika $|x - y| \neq |x' - y'|$.

# Primer – 4 kraljice i 4 skakača

**Primer 10**

*Razmotrimo sada sledeći problem: potrebno je rasporediti 4 kraljice i 4 skakača na šahovsku tablu tako da se međusobno ne napadaju.*

# Primer – 4 kraljice i 4 skakača

## Primer (nastavak)

*Slično kao i kod problema 8 kraljica, i ovde ćemo imati promenljive $x_1^Q, \ldots, x_4^Q$ (vrste za kraljice), $y_1^Q, \ldots, y_4^Q$ (kolone za kraljice), $x_1^N, \ldots, x_4^N$ (vrste za skakače) i $y_1^N, \ldots, y_4^N$ (kolone za skakače). Sve ove promenljive imaće domen $D = \{1, \ldots, 8\}$. Imaćemo sledeća ograničenja:*

- *da se kraljice međusobno ne napadaju kodiramo slično kao i ranije:* $\texttt{alldiff}(x_1^Q, \ldots, x_4^Q)$, $\texttt{alldiff}(y_1^Q, \ldots, y_4^Q)$, *kao i* $|x_i^Q - x_j^Q| \neq |y_i^Q - y_j^Q|$ *za* $i \neq j$.
- *slično kodiramo da kraljice ne napadaju skakače:* $x_i^Q \neq x_j^N$ *i* $y_i^Q \neq y_j^N$ *za svako* $i, j$, *kao i* $|x_i^Q - x_j^N| \neq |y_i^Q - y_j^N|$, *za svako* $i, j$
- *da skakači ne napadaju kraljice:* $|x_i^Q - x_j^N| \cdot |y_i^Q - y_j^N| \neq 2$, *za svako* $i, j$
- *da se skakači međusobno ne napadaju:* $|x_i^N - x_j^N| \cdot |y_i^N - y_j^N| \neq 2$, *za* $i \neq j$
- *da se nikoja dva skakača ne nalaze na istom polju:* $x_i^N \neq x_j^N \vee y_i^N \neq y_j^N$, *za* $i \neq j$

# Ograničenja proizvoda i stepena

## Definicija 7

*Ograničenje oblika*

$$y = x_1 \cdot \ldots \cdot x_n$$

*nazivamo ograničenje proizvoda. Slično, ograničenje:*

$$y = x^z$$

*nazivamo ograničenje stepena.*

## Napomena

U prethodnom primeru, ograničenje oblika $|x_1 - y_1| \cdot |x_2 - y_2| \neq c$ se može svesti na ograničenja $z_1 = x_1 - y_1$, $z_2 = x_2 - y_2$, $u_1 = |z_1|$, $u_2 = |z_2|$, $w = u_1 \cdot u_2$, $w \neq c$.

# Optimizaciona varijanta CSP problema

## Definicija 8

*Problem ograničene optimizacije* (engl. *Constrained Optimization Problem (COP)*) *je uređeni par* $(\mathcal{P}, f)$, *gde je* $\mathcal{P}$ *CSP problem nad promenljivama* $x_1, \ldots, x_n$ *sa domenima* $D_i = D(x_i)$, *a* $f : D_1 \times \ldots \times D_n \to \mathbb{R}$ *je funkcija cilja* (engl. *objective function*). *Rešenje COP problema je bilo koje rešenje CSP problema* $\mathcal{P}$ *koje maksimizuje (ili minimizuje) funkciju* $f$.

# Primer – Problem ranca

## Primer 11

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

# Primer – Problem ranca (2)

## Primer (nastavak)

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

$$s_1 x_1 + s_2 x_2 + \ldots + s_n x_n \leq K$$

*kojim se obezbeđuje da se kapacitet ranca ne prekorači. Potrebno je maksimizovati funkciju:*

$$f(x_1, \ldots, x_n) = v_1 x_1 + v_2 x_2 + \ldots + v_n x_n$$

# Primer – Golombov lenjir

## Primer 12

*Razmotrimo problem Golombovog lenjira: za date prirodne brojeve $n$ i $k$ potrebno je odrediti niz od $k$ celih brojeva $x_1 < x_2 < \ldots < x_k$ takav da su sve razlike $x_j - x_i$ (za $i < j$) međusobno različite, kao i da najveća među njima $x_k - x_1$ bude jednaka $n$.*
*NAPOMENA: U zavisnosti od $n$ i $k$, problem može da ima nula ili više rešenja.*

# Primer – Golombov lenjir (2)

## Primer (nastavak)

*Problem Golombovog lenjira se može predstaviti na sledeći način:*

- *imamo promenljive* $x_1, \ldots, x_k$ *sa domenom* $D = \{0, 1, \ldots, n\}$
- *imamo promenljive* $y_{ij}$ $(1 \leq i < j \leq k)$ *sa domenom* $D' = \{1, \ldots, n\}$
- *imamo ograničenja* $x_1 < x_2, x_2 < x_3, \ldots x_{k-1} < x_k$, $x_k - x_1 = n$
- *imamo ograničenja:* $y_{ij} = x_j - x_i$ $(1 \leq i < j \leq k)$
- *imamo ograničenje* $\texttt{alldiff}(y_{21}, y_{32}, y_{31}, \ldots, y_{k,k-1}, y_{k1})$

*NAPOMENA: iz gornjih uslova i zadatog domena, jasno je da mora biti* $x_1 = 0$ *i* $x_k = n$*. Ipak, i ova dva ograničenja se mogu zadati, iako su redundantna, jer ponekad mogu olakšati posao rešavaču.*

# Primer – Golombov lenjir, optimizaciona varijanta

## Primer 13

*Optimizaciona varijanta Golombovog lenjira: za fiksirano $k$ odrediti najmanji mogući prirodan broj $n$ takvo da problem Golombovog lenjira ima rešenje.*

*Model je isti kao ranije, s tim što bismo za veličinu domena uzeli neko $N$ dovoljno veliko da budemo sigurni da postoji rešenje (npr. $N = k \cdot k$), a umesto ograničenja $x_k - x_1 = n$ imali bismo ograničenje $x_1 = 0$. Cilj bi bio minimizovati funkciju* $f(x_1, \ldots, x_k, y_{12}, \ldots, y_{k-1,k}) = x_k$

# Primer – svi intervali

## Primer 14

*Da li je moguće odsvirati svih 12 tonova jedne oktave (c, c#, d, d#, itd.) u takvom redosledu da nikoja dva intervala između susednih odsviranih tonova ne budu jednaka?*

*NAPOMENA: ako tonove redom označimo brojevima* $0, 1, \ldots, 11$*, tada je interval između dva tona* $x$ *i* $y$ *jednak njihovoj apsolutnoj razlici* $|x - y|$ *(1 je poluton, a 11 je velika sedmica). Problem se sada može formulisati i ovako:*

*Da li postoji permutacija* $x_0 x_1 \ldots x_{11}$ *brojeva* $0, 1, \ldots, 11$*, takva da za* $y_i = |x_i - x_{i-1}|$ *niz brojeva* $y_1 y_2 \ldots y_{11}$ *predstavlja permutaciju brojeva* $1, 2, \ldots, 11$*?*

# Primer – svi intervali (2)

## Primer (nastavak)

*Označimo promenljive kao na prethodnom slajdu. Domen $x$ promenljivih će biti skup $\{0, 1, \ldots, 11\}$, a domen $y$ promenljivih će biti $\{1, 2, \ldots, 11\}$. Sada imamo ograničenja:*

- $y_i = |x_i - x_{i-1}|$, *za* $i \in \{1, \ldots, 11\}$
- $\texttt{alldiff}(x_0, \ldots, x_{11})$
- $\texttt{alldiff}(y_1, \ldots, y_{11})$

# Primer – Langfordovi parovi

## Primer 15

*Za dato $n$, formirati permutaciju brojeva $1, 1, 2, 2, \ldots, n, n$ takvu da između dve jedinice postoji jedan broj, između dve dvojke dva broja, i sl. Na primer, za $n = 4$, jedna moguća permutacija bila bi $4, 1, 3, 1, 2, 4, 3, 2$.*

*Problem nema rešenje za svako $n$, dok za neke vrednosti $n$ može imati i više različitih rešenja.*

# Primer – Langfordovi parovi (2)

## Primer (nastavak)

*Uvedimo promenljive $x_1^1$, $x_1^2$ (pozicije prve i druge kopije jedinice u nizu), $x_2^1$, $x_2^2$ (pozicije prve i druge kopije dvojke u nizu), i sl. Sve ove promenljive (njih $2n$) imaju domen $D = \{1, 2, \ldots, 2n\}$. Pritom, važe ograničenja:*

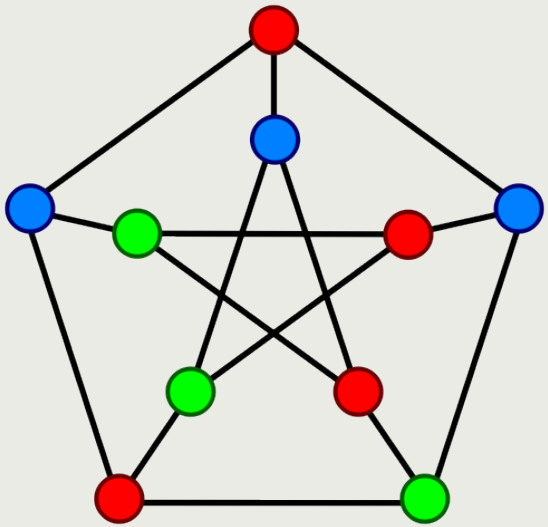
- $\texttt{alldiff}(x_1^1, x_1^2, x_2^1, x_2^2, \ldots, x_n^1, x_n^2)$ *(svi moraju biti na različitim pozicijama)*
- $x_1^2 - x_1^1 = 2$, $x_2^2 - x_2^1 = 3$, $\ldots$, $x_n^2 - x_n^1 = n + 1$

*Primetimo da ne koristimo apsolutne vrednosti, npr. $|x_1^2 - x_1^1| = 2$, jer želimo da razbijemo simetrije (zahtevamo da druga kopija bude desno od prve).*

# Primer – bojenje grafa

## Primer 16

*Obojiti čvorove grafa pomoću n različitih boja tako da nikoja dva susedna čvora ne budu obojena istom bojom.*
*Na sledećoj slici dat je primer ispravno obojenog grafa sa 3 boje.*



*NAPOMENA: Za $n \geq 3$ problem bojenja grafa je NP-kompletan.*

# Primer – bojenje grafa (2)

## Primer (nastavak)

*Problem bojenja grafa se može jednostavno predstaviti kao CSP problem, tako što se uvede po jedna promenljiva za svaki čvor, sa domenom* $\{1, 2, \ldots, n\}$ *(intuitivno, vrednost te promenljive predstavlja boju čvora). Sada treba uvesti ograničenja različitosti* $x_i \neq x_j$ *za svaka dva susedna čvora* $i$ *i* $j$*.*

*Alternativno, ako znamo da neki skup čvorova* $\{x_{i_1}, x_{i_2}, \ldots, x_{i_k}\}$ *čine kliku*[1] *veličine* $k$*, tada uslove različitosti među njima možemo kompaktno predstaviti ograničenjem* $\texttt{alldiff}(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$.

[1] Klika je skup čvorova grafa takav da između svaka dva čvora postoji grana

# Primer – maksimalna klika

**Primer 17**

*Razmotrimo problem maksimalne klike: u datom grafu pronaći kliku maksimalne veličine.*

# Primer – maksimalna klika (2)

## Primer (nastavak)

*Za svaki od čvorova grafa $v_1, \ldots, v_n$ uvedimo po jednu promenljivu $x_1, \ldots, x_n$ sa domenom $\{0, 1\}$. Intuitivno, $x_i = 1$ akko je $v_i$ član maksimalne klike. Sada imamo ograničenja oblika $x_i \neq 1 \vee x_j \neq 1$ za svaka dva čvora $v_i$ i $v_j$ za koje ne postoji grana između njih. Cilj je maksimizovati funkciju:*

$$f(x_1, \ldots, x_n) = x_1 + x_2 + \ldots + x_n$$

# Disjunktivno ograničenje i ograničenje negacije

## Definicija 9

*Ograničenje oblika $C_1 \vee \ldots \vee C_k$, gde su $C_1, \ldots, C_k$ takođe ograničenja nazivamo disjunktivno ograničenje.*
*Ograničenje oblika $\neg C$ gde je $C$ ograničenje nazivamo ograničenje negacije.*

## Napomena

- Disjunktivno ograničenje i ograničenje negacije su zapravo konstruktori kojima se kreiraju nova ograničenja od postojećih ograničenja
- Drugi logički veznici poput implikacije i ekvivalencije se mogu svesti na ova ograničenja:
  - $C_1 \Rightarrow C_2$ se svodi na $\neg C_1 \vee C_2$
  - $C_1 \Leftrightarrow C_2$ se svodi na dva ograničenja: $\neg C_1 \vee C_2$ i $C_1 \vee \neg C_2$
  - *if* $C$ *then* $C_1$ *else* $C_2$ se svodi na ograničenja: $\neg C \vee C_1$ i $C \vee C_2$

# Primer – sudoku

## Primer 18

*Problem sudoku slagalice je uopštenje problema latinskog kvadrata, sa dodatnim uslovima različitosti po ,,kvadratima". Primer jedne sudoku slagalice dat je na sledećoj slici.*

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

*U formi CSP problema, imaćemo po jednu promenljivu $x_{ij}$ za svako polje $(i, j)$ sa domenom $\{1, 2, \ldots, 9\}$, kao i ograničenja* `alldiff` *po vrstama, kolonama i $3 \times 3$ kvadratima. Takođe, imaćemo jednakosti koje predstavljaju zadate vrednosti (npr. $x_{11} = 5$).*

# Primer – društveni golferi

## Primer 19

*Grupa od $m \cdot n$ igrača jednom nedeljno igra golf. Dele se u $m$ ekipa po $n$ igrača. Kako bi se što više družili, cilj im je da svaki put menjaju ekipe, tako da nikoja dva igrača ne igraju dva puta u istom timu. Potrebno je napraviti takav raspored za $p$ nedelja. Da li je to moguće?*

*Odgovor na ovo pitanje zavisi od parametara $m, n, p$. U pitanju je težak kombinatorni problem iz klase problema raspoređivanja.*

# Primer – društveni golferi (2)

## Primer (nastavak)

*Problem društvenih golfera se može modelovati kao CSP problem tako što najpre pretpostavimo da su igrači unapred numerisani brojevima* $0, 1, \ldots, mn-1$*, a zatim uvedemo promenljivu* $x_{ijk}$ *(*$i \in \{1, \ldots, m\}$*,* $j \in \{1, \ldots, n\}$*,* $k \in \{1, \ldots, p\}$*) sa domenom* $\{0, 1, \ldots, mn-1\}$*. Intuitivno, vrednost promenljive* $x_{ijk}$ *označava igrača koji k-te nedelje igra u i-toj ekipi na j-toj poziciji. Sada ćemo za svako fiksirano k imati ograničenje* `alldiff` *po svim* $x_{ijk}$ *za* $i \in \{1, \ldots, m\}$*,* $j \in \{1, \ldots, n\}$ *(jer jedan isti igrač ne može biti na dve različite pozicije u isto vreme). U cilju razbijanja simetrija, pretpostavićemo da je za svako fiksirano i i k (tj. fiksiranu ekipu u fiksiranoj nedelji) imamo* $x_{i1k} < x_{i2k} < \ldots < x_{ink}$ *(tj. ne želimo da razmatramo različite permutacije iste ekipe).*

| | *Ekipa 1* | *Ekipa 2* | $\ldots$ | *Ekipa m* |
|---|---|---|---|---|
| *Nedelja 1* | *1 2 3 4* | *5 6 7 8* | $\ldots$ | |
| *Nedelja 2* | *1 5 9 13* | *2 6 10 14* | $\ldots$ | |
| $\vdots$ | $\vdots$ | $\vdots$ | $\ddots$ | |
| *Nedelja p* | | | | |

# Primer – društveni golveri (3)

## Primer (nastavak)

*Dalje, potrebno je obezbediti da nikoja dva igrača ne igraju više od jednom u istoj ekipi. Da bismo to postigli, primetimo bijektivno mapiranje:*

$$(u, v) \mapsto mn \cdot u + v$$

*skupa* $\{0, 1, \ldots, mn-1\} \times \{0, 1, \ldots, mn-1\}$ *na skup* $\{0, 1, \ldots, m^2n^2-1\}$. *Intuitivno, ovo mapiranje svakom uređenom paru igrača pridružuje jedinstven kôd. Sada uvodimo promenljive* $y_{ij_1j_2k}$ *(za* $i \in \{1, \ldots, m\}$, $k \in \{1, \ldots, p\}$ *i* $1 \leq j_1 < j_2 \leq n$*) sa domenom* $\{0, 1, \ldots, m^2n^2-1\}$ *i zadajemo ograničenja* $y_{ij_1j_2k} = mn \cdot x_{ij_1k} + x_{ij_2k}$ *(tj. promenljiva* $y_{ij_1j_2k}$ *predstavlja kôd pridružen paru igrača koji igraju u i-toj ekipi na pozicijama* $j_1$ *i* $j_2$ *tokom k-te nedelje). Sada imamo jedno* `alldiff` *ograničenje nad svim y promenljivama (jer se ni jedan kôd ne sme pojaviti više od jednom, što bi značilo da se isti par igrača dva puta pojavio zajedno u ekipi).*

# Primer – zebra

## Primer 20

*Imamo 5 kuća, jednu do druge: crvena, zelena, žuta, bela i plava, u nekom redosledu. Takođe, imamo 5 osoba: Englez, Španac, Ukrajinac, Norvežanin i Japanac koji žive u tim kućama (svako u svojoj). Ove osobe piju 5 različitih napitaka: kafu, čaj, mleko, orandžadu i limunadu. Takođe, svaka osoba poseduje jednog od sledećih kućnih ljubimaca: konja, psa, lisicu, zebru i puža (opet, svako ima različitog ljubimca). Najzad, svaka od osoba se bavi jednim od sledećih zanimanja (svaka različitim): programer, pevač, pekar, advokat i profesor. Ono što znamo je sledeće:*

- *Englez živi u crvenoj kući.*
- *Španac poseduje psa.*
- *Kafu pije vlasnik zelene kuće.*
- *Ukrajinac pije čaj.*
- *Zelena kuća je odmah iza bele kuće.*
- *Programer poseduje puža.*
- *Pevač živi u žutoj kući.*
- *Mleko se pije u srednjoj kući.*
- *Norvežanin živi u prvoj kući.*
- *Pekar živi u kući pored kuće čoveka koji ima lisicu.*
- *Pevač živi u kući pored kuće čoveka koji ima konja.*
- *Advokat pije orandžadu.*
- *Japanac je profesor.*
- *Norvežanin živi pored plave kuće.*

*Pitanje je: ko poseduje zebru?*

# Primer – zebra (2)

## Primer (nastavak)

*Jedno moguće rešenje se može predstaviti tabelom:*

| **Kuća** | **1** | **2** | **3** | **4** | **5** |
|---|---|---|---|---|---|
| **Boja** | *žuta* | *plava* | *crvena* | *bela* | *zelena* |
| **Nacionalnost** | *Norvežanin* | *Ukrajinac* | *Englez* | *Španac* | *Japanac* |
| **Napitak** | *limunada* | *čaj* | *mleko* | *orandžada* | *kafa* |
| **Zanimanje** | *pevač* | *pekar* | *programer* | *advokat* | *profesor* |
| **Ljubimac** | *lisica* | *konj* | *puž* | *pas* | **zebra** |

# Primer – zebra (3)

## Primer (nastavak)

*Da bismo ovaj problem predstavili kao CSP, uvedimo promenljive zuta, plava, . . . , Englez, Norvezanin,. . . , limunada, caj, . . . , pevac, pekar, . . . , lisica, konj, . . . , pri čemu svaka od ovih promenljivih ima domen* $\{1, 2, \ldots, 5\}$. *Intuitivno, vrednost svake promenljive predstavlja redni broj kuće za koju se vezuje odgovarajuća osoba, profesija, ljubimac, i sl. Sada se ograničenja mogu lako izraziti:*

- $Englez = crvena$
- $Spanac = pas$
- $kafa = zelena$
- $Ukrajinac = caj$
- $zelena = bela + 1$
- $programer = puz$
- $pevac = zuta$
- $mleko = 3$
- $Norvezanin = 1$
- $|pekar - lisica| = 1$
- $|pevac - konj| = 1$
- $advokat = orandzada$
- $Japanac = profesor$
- $|Norvezanin - plava| = 1$

*Dodatno, imamo ograničenja* $\texttt{alldiff}(Englez, \ldots, Japanac)$, $\texttt{alldiff}(zuta, \ldots, bela)$, $\texttt{alldiff}(pevac, \ldots, profesor)$, $\texttt{alldiff}(kafa, \ldots, limunada)$ *i* $\texttt{alldiff}(pas, \ldots, zebra)$, *kako bismo obezbedili različitost.*

# Primer – Neptunove sluge

## Primer 21

*Među Neptunovim slugama nalaze se oktopodi sa po 6, 7 ili 8 nogu. Oni koji imaju 7 nogu uvek lažu, dok oni koji imaju 6 ili 8 nogu uvek govore istinu. Četiri oktopoda, Plavi, Crveni, Zeleni i Žuti su se jednom prilikom srela. Tada je Plavi izjavio: ,,Nas četvorica zajedno imamo 28 nogu". Zeleni je izjavio: ,,Nas četvorica zajedno imamo 27 nogu". Žuti je izjavio: ,,Nas četvorica zajedno imamo 26 nogu". Crveni je izjavio: ,,Nas četvorica zajedno imamo 25 nogu". Koliko nogu ima koji oktopod?*

# Primer – Neptunove sluge (2)

## Primer (nastavak)

*Označimo sa $B$, $R$, $G$, $Y$ redom promenljive koje označavaju broj nogu Plavog, Crvenog, Zelenog i Žutog oktopoda. Svaka od ovih promenljivih ima domen $\{6, 7, 8\}$. Neka je $S$ ukupan broj nogu svih oktopoda (ova promenljiva može imati domen $\{24, \ldots, 32\}$). Pored ograničenja $S = B + R + G + Y$, imaćemo i ograničenja:*

- $B = 7 \Leftrightarrow S \neq 28$
- $R = 7 \Leftrightarrow S \neq 25$
- $G = 7 \Leftrightarrow S \neq 27$
- $Y = 7 \Leftrightarrow S \neq 26$

# Primer – raspored časova

## Primer 22

*Posmatrajmo školu koja ima 40 odeljenja, pri čemu svako odeljenje pohađa 35 časova nedeljno, raspoređenih u pet radnih dana (7 časova dnevno). Svakoga dana postoji tačno sedam 45-minutnih termina na koje je potrebno rasporediti časove. Neki časovi se moraju držati u blokovima od po 2 ili 3 susedna časa istog dana. Nastavu drži određeni broj nastavnika, pri čemu svaki od njih ima od 14 do 20 časova nedeljno. Svaki od časova svakog odeljenja je unapred dodeljen konkretnom nastavniku. Zadatak je formirati raspored časova.*

# Primer – raspored časova (2)

## Primer (nastavak)

*Numerišimo najpre sve termine redom brojevima* $1, 2, \ldots, 35$*:*

| Ponedeljak | Utorak | Sreda | Četvrtak | Petak |
|---|---|---|---|---|
| 1 | 8 | 15 | 22 | 29 |
| 2 | 9 | 16 | 23 | 30 |
| 3 | 10 | 17 | 24 | 31 |
| 4 | 11 | 18 | 25 | 32 |
| 5 | 12 | 19 | 26 | 33 |
| 6 | 13 | 20 | 27 | 34 |
| 7 | 14 | 21 | 28 | 35 |

# Primer – raspored časova (3)

## Primer (nastavak)

*Numerišimo, dalje, nastavnike, grupe i njihove časove. Na primer:*

| **Nastavnik** | **Kôd** |
|---|---|
| *Petar Petrović* | *1* |
| *Dejan Mikić* | *2* |
| ⋮ | ⋮ |
| *Jovan Jovanović* | *12* |

| **Odeljenje** | **Kôd** |
|---|---|
| $I_1$ | *1* |
| $I_2$ | *2* |
| ⋮ | ⋮ |
| $IV_{10}$ | *40* |

| **Čas odeljenja $I_1$** | **Kôd** |
|---|---|
| *Matematika* | *1* |
| *Matematika* | *2* |
| *Matematika* | *3* |
| *Matematika* | *4* |
| *Srpski* | *5* |
| *Srpski* | *6* |
| *Srpski* | *7* |
| *Fizičko* | *8* |
| *Fizičko* | *9* |
| ⋮ | ⋮ |
| *Filozofija* | *35* |

# Primer – raspored časova (4)

## Primer (nastavak)

*Pretpostavimo da je dodela časova nastavnicima, zajedno sa uslovima za blokove data u sledećoj tabeli:*

| Kôd odeljenja | Kôd časa | Kôd nastavnika | Blok |
|---|---|---|---|
| 1 | 1 | 1 | 1 |
| 1 | 2 | 1 | 1 |
| 1 | 3 | 1 | 2 |
| 1 | 4 | 1 | 2 |
| 1 | 5 | 2 | 3 |
| 1 | 6 | 2 | 3 |
| 1 | 7 | 2 | 3 |
| 1 | 8 | 7 | 4 |
| 1 | 9 | 7 | 5 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 1 | 35 | 12 | 16 |

# Primer – raspored časova (5)

## Primer (nastavak)

*Da bismo opisani problem predstavili kao CSP problem, uvedimo promenljive $x_{ijk}$ za i-tog nastavnika koji drži k-ti čas j-tom odeljenju. Domen ovih promenljivih biće skup $\{1, \ldots, 35\}$, a vrednost promenljive $x_{ijk}$ intuitivno predstavlja redni broj termina u kome se taj čas održava.*

- *da bismo obezbedili da i-ti nastavnik ne drži dva časa u isto vreme, zadajemo ograničenje* `alldiff` *nad svim $x_{ijk}$ promenljivama za to fiksirano i*
- *da bismo obezbedili da j-ta grupa nema dva časa u isto vreme, zadajemo ograničenje* `alldiff` *nad svim $x_{ijk}$ za to fiksirano j*
- *da bismo izrazili uslove vezane za blokove, za svaka dva časa $k_1$ i $k_2$ koji bi trebalo da budu susedni u bloku zadajemo ograničenje $x_{ijk_1} + 1 = x_{ijk_2}$*
- *kako se blokovi ne bi ,,razbili" u dva dana, ako npr. k-ti čas nije poslednji čas u svom bloku, dodajemo uslov da on ne može biti poslednji čas nekog dana: $x_{ijk} \bmod 7 \neq 0$.*

# Ograničenja *div* i *mod*

## Definicija 10

*Neka su* $x, y, z$ *promenljive. Ograničenja:*

$$x = y \ div \ z$$

*odnosno*

$$x = y \ mod \ z$$

*nazivamo ograničenjima celobrojnog količnika i ostatka pri deljenju, respektivno. Umesto promenljive* $z$ *može da stoji i konstanta* $c$*.*

# Nizovi i matrice

## Nizovske promenljive

- Promenljive nizovskog (ili matričnog) tipa se veoma često javljaju u modelima
  - strogo govoreći, promenljiva nizovskog tipa ima domen $\underbrace{D \times \ldots \times D}_{n}$, gde je $D$ domen elementa niza, a $n$ dužina niza
  - izraz $x[i]$ označava pristup $i$-toj komponenti odgovarajuće $n$-torke
  - višedimenzioni nizovi (matrice) se svode na jednodimenzione: $x[i,j] \rightsquigarrow x[n \cdot i + j]$
- Pristup elementu $n$-torke ($x[i]$) se formalizuje uvođenjem ograničenja element

## Definicija 11

*Ograničenje* $\texttt{element}(y, x, v)$*, gde je* $x$ *nizovska promenljiva (dužine* $n$*) a* $y, v$ *su skalarne promenljive (ili konstante), se definiše na sledeći način:*

$$\texttt{element}(y, x, v) = \{(i, (d_1, \ldots, d_n), d_i) \mid 1 \leq i \leq n\}$$

## Napomena

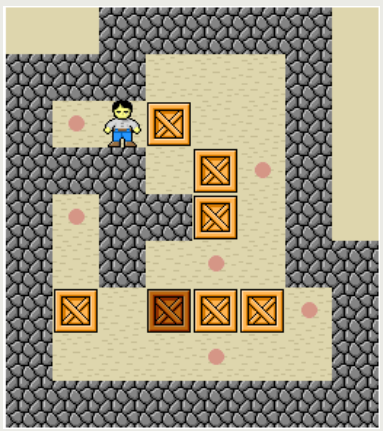
Intuitivno, $\texttt{element}(y, x, v)$ označava da je $v = x[y]$.

- primetimo da $y$ može biti konstanta ili promenljiva

# Primer – Sokoban

## Primer 23

*Sokoban je ,,čuvar skladišta" koji ima zadatak da napakuje sanduke na obeležena polja. Pritom, on ne može prolaziti kroz zidove i može gurati samo po jedan sanduk (ne može vući sanduke niti gurati više od jednog sanduka odjednom).*



*Pitanje: za dato N, da li je moguće rešiti dati Sokoban problem u N koraka?*

# Primer – Sokoban (2)

## Primer (nastavak)

*Pretpostavimo da se tabla za igru predstavlja matricom $m \times n$. Na svakom polju table se nalazi ili zid, ili odredište, ili ni jedno od ta dva. Ova konfiguracija je fiksirana i ne menja se tokom igre. Imamo sledeće promenljive (tj. matrice promenljivih):*

- $wall[i,j]$ *sa domenom* $\{true, false\}$ *koja označava da li se na poziciji* $(i,j)$ *nalazi zid*
- $home[i,j]$ *sa domenom* $\{true, false\}$ *koja označava da li je na poziciji* $(i,j)$ *odredište*

*Ove promenljive imaju unapred zadate vrednosti (ograničenjima oblika npr.* $wall[i,j] = true$ *ili* $home[i,j] = false$*).*
*Ako se na polju ne nalazi zid, tada se na tom polju može nalaziti ili kutija, ili Sokoban, ili ni jedno od ta dva. Imamo sledeće promenljive:*

- $box_k[i,j]$ *sa domenom* $\{true, false\}$ *koja označava da se nakon* $k$ *koraka na poziciji* $(i,j)$ *nalazi kutija* $(k \in \{0, \ldots, N\})$
- $I_k$ *sa domenom* $\{1, 2, \ldots, m\}$ *i* $J_k$ *sa domenom* $\{1, 2, \ldots, n\}$ *koje označavaju, respektivno, redni broj vrste i kolone u kojoj se nalazi Sokoban nakon* $k$*-tog koraka* $(k \in \{0, \ldots, N\})$

*Najzad, imamo promenljivu* $move_k$ *sa domenom* $\{up, down, left, right\}$ *koja označava potez u* $k$*-tom koraku* $(k \in \{1, \ldots, N\})$*.*

# Primer – Sokoban (3)

## Primer (nastavak)

*Početno stanje igre kodiramo uslovima nad promenljivama* $box_0[i, j]$, $K_0$ *i* $J_0$. *Na primer, u našem primeru sa slike imamo da se Sokoban na početku nalazi na poziciji* $(3, 3)$, *pa ćemo imati ograničenja* $I_0 = 3$ *i* $J_0 = 3$. *Slično, na polju* $(3, 4)$ *imamo kutiju, pa ćemo imati uslov* $box_0[3, 4] = true$, *i td.*

# Primer – Sokoban (4)

## Primer (nastavak)

*Da bismo garantovali da se odgovarajući potez može primeniti u k-tom koraku, moramo zadati ograničenja za tekuće $(k-1)$-vo stanje. Na primer, za potez ,,na gore'':*

$$move_k = up \Rightarrow \left( \begin{array}{c} l_{k-1} > 1 \wedge \\ wall[l_{k-1} - 1, J_{k-1}] = false \wedge \\ box_{k-1}[l_{k-1} - 1, J_{k-1}] = true \Rightarrow \\ l_{k-1} > 2 \wedge \\ box_{k-1}[l_{k-1} - 2, J_{k-1}] = false \wedge \\ wall[l_{k-1} - 2, J_{k-1}] = false \end{array} \right)$$

*Intuitivno, da bismo mogli da idemo na gore, ne smemo biti u prvoj vrsti, u vrsti iznad ne sme biti zid, a ako je iznad nas kutija, tada i iznad nje ne sme biti ni zid ni kutija.*

# Primer – Sokoban (5)

## Primer (nastavak)

*Da bismo opisali efekat k-tog koraka, moramo uspostaviti odnos između tekućeg $(k-1)$-vog i narednog k-tog stanja. Ovaj efekat zavisi od poteza koji je izvršen u k-tom koraku. Na primer, za potez ,,na gore", imamo sledeće ograničenje:*

$$move_k = up \Rightarrow \begin{array}{c} I_k = I_{k-1} - 1 \wedge \\ J_k = J_{k-1} \wedge \\ \left( \begin{array}{c} box_{k-1}[I_{k-1}-1, J_{k-1}] = false \Rightarrow \\ \forall\ i,j.\ box_k[i,j] = box_{k-1}[i,j] \end{array} \right) \wedge \\ \left( \begin{array}{c} box_{k-1}[I_{k-1}-1, J_{k-1}] = true \Rightarrow \\ \forall\ i,j.\ box_k[i,j] = box'_{k-1}[i,j] \end{array} \right) \end{array}$$

*pri čemu je sa $box'_{k-1}[i,j]$ označena matrica koja se poklapa sa $box_{k-1}[i,j]$ na svim pozicijama osim što su vrednosti na pozicijama $(I_{k-1}-1, J_{j-1})$ i $(I_{k-1}-2, J_{k-1})$ razmenjene (ovaj uslov treba raspisati, ali to ovde ne činimo zbog nedostatka prostora). Intuitivno, ako je k-ti potez ,,na gore", tada se Sokoban u sledećem stanju pomera za jednu vrstu na gore (ostaje u istoj koloni), pri čemu ako iznad nije bilo kutije, sve kutije ostaju na svojim mestima, a u suprotnom se ta jedna kutija pomera za jednu poziciju na gore.*

# Primer – Sokoban (6)

## Primer (nastavak)

*Najzad, potrebno je da nakon $N$ koraka sve kutije budu na svom mestu:*

$$\forall\ i, j.\ box_N[i, j] = home[i, j]$$

## Napomena

Zapis $\forall\ i, j.C(i, j)$ označava da se radi o skupu ograničenja $C(i, j)$ dobijenih iteracijom $i$ i $j$ kroz skupove $\{1, \ldots, m\}$ i $\{1, \ldots, n\}$, respektivno. U većini jezika za modelovanje, ovo ,,raspisivanje" se obavlja automatski.

# Primer – raspored časova (ponovo)

## Primer 24

*Pretpostavimo dodatni uslov: nastavnici ne vole da ustaju rano, pa želimo da obezbedimo da svaki nastavnik najviše dva puta nedeljno ima prvi čas. Uvedimo najpre promenljivu* $c_{ijk}$ *koja označava redni broj časa u toku dana kada i-ti nastavnik drži k-ti čas j-tom odeljenju. Ovo obezbeđujemo nametanjem ograničenja* $c_{ijk} = (x_{ijk} - 1) \bmod 7 + 1$. *Sada možemo zadati ograničenja:*

$$\texttt{atmost}(c_{ij_1k_1}, c_{ij_2k_2}, \ldots, 2, 1)$$

*čime se zahteva da, za svako fiksirano* $i$, *od svih promenljivih* $c_{ijk}$ *najviše dve imaju vrednost* 1.

# Ograničenja atleast i atmost

## Definicija 12

*Ograničenje* $\texttt{atleast}(x_1, \ldots, x_n, l, c)$, *gde su* $x_1, \ldots, x_n$ *promenljive, a* $l$ *i* $c$ *konstante, definišemo na sledeći način:*

$$\texttt{atleast}(x_1, \ldots, x_n, l, c) = \{(d_1, \ldots, d_n) \mid |\{i \mid d_i = c\}| \geq l\}$$

*Slično, ograničenje* $\texttt{atmost}(x_1, \ldots, x_n, u, c)$ *definišemo na sledeći način:*

$$\texttt{atmost}(x_1, \ldots, x_n, u, c) = \{(d_1, \ldots, d_n) \mid |\{i \mid d_i = c\}| \leq u\}$$

## Napomena

Ograničenja kardinalnosti nameću gornju, odnosno donju, granicu za broj promenljivih koje uzimaju neku zadatu vrednost.

# Primer – raspored časova (ponovo)

## Primer 25

*Pretpostavimo da, dodatno, želimo da osiguramo da svaki nastavnik svaki dan ima bar po jedan čas. Slično kao i malopre, uvedimo promenljivu* $d_{ijk}$ *koja označava dan (*1 *za ponedeljak,* 5 *za petak) kada* $i$*-ti nastavnik drži* $k$*-ti čas* $j$*-tom odeljenju. Ovo obezbeđujemo ograničenjima* $d_{ijk} = (x_{ijk} - 1)$ *div* $7 + 1$*. Sada možemo nametnuti ograničenja:*

$$\begin{aligned} &atleast(d_{ij_1k_1}, d_{ij_2k_2}, \ldots, 1, 1)\\ &atleast(d_{ij_1k_1}, d_{ij_2k_2}, \ldots, 1, 2)\\ &\vdots\\ &atleast(d_{ij_1k_1}, d_{ij_2k_2}, \ldots, 1, 5) \end{aligned}$$

*Drugim rečima, bar jedan čas je raspoređen u ponedeljak, bar jedan u utorak, i td.*

# Ograničenje globalne kardinalnosti

## Primer (nastavak)

*Alternativno, umesto pojedinačnih* `atleast` *ograničenja, možemo nametnuti jedno ograničenje:*

$$\mathrm{gcc}(d_{ij_1k_1}, d_{ij_2k_2}, \ldots, \{(1,1,7),(2,1,7),\ldots,(5,1,7)\})$$

## Definicija 13

*Ograničenje globalne kardinalnosti* gcc *definišemo na sledeći način:*

$$\mathrm{gcc}(x_1,\ldots,x_n,\left\{\begin{array}{l}(c_1,l_1,u_1),\\ \ldots,\\ (c_k,l_k,u_k)\end{array}\right\}) = \{(d_1,\ldots,d_n) \mid \forall j.l_j \leq |\{d_i \mid d_i = c_j\}| \leq u_j\}$$

*Dakle, za svaku vrednost $c_j$ zahtevamo da broj promenljivih koje uzimaju tu vrednost pripada intervalu $[l_j, u_j]$.*

# Primer – raspoređivanje procesa

## Primer 26

*Razmotrimo problem raspoređivanje poslova: imamo m izvršioca i n poslova koje treba završiti. Svaki posao ima svoje trajanje. Postoje međuzavisnosti između poslova – neki poslovi ne mogu početi pre nego što se neki drugi završe. Takođe, imamo r različitih (obnovljivih) resursa, pri čem svaki resurs ima ograničenu količinu. Za svaki posao se zna kolika mu je količina kog resursa potrebna. Potrebno je rasporediti poslove na m izvršioca tako da ukupno vreme za završetak svih poslova bude minimalno. Pritom, ni u jednom trenutku ne sme biti prekoračena maksimalna količina svakog resursa.*

# Primer – raspoređivanje procesa (2)

## Primer (nastavak)

*Pretpostavimo da je trajanje izraženo u celim vremenskim jedinicama (npr. u satima). Neka je $l_i$ konstanta koja predstavlja trajanje i-tog posla. Neka promenljiva $x_i$ označava vremensku jedinicu kada počinje i-ti posao da se izvršava. Ove promenljive mogu imati domen $\{1, \ldots, N\}$, gde je N jednako zbiru dužina svih poslova (time je garantovano da će biti rešenja, makar se izvršavali sekvencijalno). Neka promenljiva $y_{ij}$ (sa domenom $\{0, 1\}$) ima vrednost 1 akko i-ti posao traje tokom j-tog sata. Ovo postižemo ograničenjem: $y_{ij} = 1 \Leftrightarrow x_i \leq j \leq x_i + l_i - 1$. Sada ćemo, za svako $j \in \{1, \ldots N\}$, imati ograničenje:*

$$y_{1j} + y_{2j} + \ldots + y_{nj} \leq m$$

*kako bismo obezbedili da najviše m poslova budu aktivni u svakom trenutku. Slično, ako za svaki resurs k njegov limit označimo sa $L_k$, a količinu k-tog resursa koja je potrebna za i-ti posao označimo sa $R_{ik}$, imaćemo ograničenja (za svako $j, k$):*

$$R_{1k} \cdot y_{1j} + \ldots + R_{nk} \cdot y_{nj} \leq L_k$$

*Najzad, za svaka dva posla $i_1$ i $i_2$ takvih da $i_2$ zavisi od $i_1$, imaćemo ograničenje: $x_{i_2} \geq x_{i_1} + l_{i_1}$. Da bismo optimizovali ukupno vreme izvršavanja poslova, uvešćemo i promenljivu D, uz ograničenja: $x_i + l_i - 1 \leq D$, za svako i. Sada je zadatak minimizovati funkciju:*

$$f(x_1, \ldots, x_n, y_{11}, \ldots, y_{nN}, D) = D$$

# Ograničenje `cumulative`

## Definicija 14

*Ograničenje* `cumulative` *definiše se na sledeći način:*

$$\texttt{cumulative}(x_1, \ldots, x_n, \left\{ \begin{array}{c} (l_1, r_i), \\ \ldots, \\ (l_n, r_n) \end{array} \right\}, m) =$$
$$\left\{ (d_1, \ldots, d_n) \mid \forall t. \sum_{i \mid d_i \leq t < d_i + l_i} r_i \leq m \right\}$$

## Napomena

Intuitivno, promenljiva $x_i$ predstavlja vreme početka $i$-tog posla, a konstante $l_i$ i $r_i$ trajanje $i$-tog posla i količinu datog resursa koja je potrebna za $i$-ti posao. Konstanta $m$ predstavlja ukupnu količinu datog resursa. Ograničenje obezbeđuje da u svakom trenutku upotreba datog resursa ne prekorači navedeni limit.

# Primer – raspoređivanje procesa (3)

## Primer (nastavak)

*Alternativno, umesto uvođenja promenljivih $y_{ij}$ mogli smo uslov za ograničavanje broja aktivnih poslova u svakoj vremenskoj jedinici izraziti na sledeći način:*

$$\texttt{cumulative}(x_1, \ldots, x_n, \{(l_1, 1), \ldots, (l_n, 1)\}, m)$$

*Slično, uslove za svaki pojedinačan resurs mogli smo predstaviti ograničenjima oblika:*

$$\texttt{cumulative}(x_1, \ldots, x_n, \{(l_1, R_{1k}), \ldots, (l_n, R_{nk})\}, L_k)$$

*za* $k \in \{1, \ldots, r\}$.

# Pregled

# CSP rešavači

## CSP rešavači

Programske implementacije procedura za rešavanje CSP problema nazivaju se CSP rešavači:

- prihvataju na ulazu CSP (ili COP) problem predstavljen u nekom jeziku za modelovanje (najpoznatiji je MiniZinc)
- kao rezultat daju pronađeno rešenje, ako postoji
- rešavači za probleme nad konačnim domenima – FD CSP rešavači (engl.Finite Domain)
- neki poznatiji FD CSP rešavači: Choco, Mistral, chuffed, Gecode, minion
- u nastavku razmatramo principe na kojima su zasnovani FD CSP rešavači

# Rešeni i neuspešni problemi

## Rešeni i neuspešni problemi

U cilju rešavanja CSP problema, razmatramo najpre dva trivijalna slučaja: rešene i neuspešne probleme.

## Definicija 15

*CSP problem $\mathcal{P}$ nad nizom promenljivih $(x_1, \ldots, x_n)$ je rešen (engl. solved) ako je njegov skup rešenja jednak $D_1 \times \ldots \times D_n$, gde je $D_i = D(x_i)$ i ni jedan $D_i$ nije prazan.*

## Definicija 16

*CSP problem $\mathcal{P}$ je neuspešan (engl. failed) ukoliko je domen bilo koje njegove promenljive prazan skup, ili je bilo koje njegovo ograničenje prazan skup.*

## Rešavanje ova dva specijalna tipa problema je trivijalno:

- skup rešenja rešenog CSP problema je neposredno dat domenima promenljivih, pa nije potrebno nikakvo dalje rešavanje
  - rešen problem je uvek konzistentan, jer se podrazumeva da su domeni neprazni
- skup rešenja neuspešnog problema je očigledno prazan, te znamo da je problem nekonzistentan

## Opšta ideja rešavanja CSP problema

Cilj je svesti dati CSP problem na jedan ili više ekvivalentnih rešenih i/ili neuspešnih problema

# Atomični problemi

## Definicija 17

*Za CSP problem $\mathcal{P}$ kažemo da je atomičan, ako su mu svi domeni jednočlani ili prazni.*

## Napomena

Ako je problem $\mathcal{P}$ atomičan i bez praznih domena, pri čemu jedina $n$-torka elemenata iz domena $(d_1, \ldots, d_n)$ zadovoljava sva ograničenja problema (što je lako proveriti), tada je $\mathcal{P}$ rešen, a u suprotnom je neuspešan.

- Odavde sledi da do rešenih/neuspešnih problema možemo stići tako što težimo ka atomičnim problemima
- Ključno pitanje je: kako?

# Ekvivalentni problemi

## Definicija 18

*Dva CSP problema $\mathcal{P}_1$ i $\mathcal{P}_2$ definisana nad istim nizom promenljivih $\mathcal{X}$ su ekvivalentna, u oznaci $\mathcal{P}_1 \equiv \mathcal{P}_2$, ako im se skupovi rešenja poklapaju, tj. ako je svako rešenje problema $\mathcal{P}_1$ ujedno i rešenje problema $\mathcal{P}_2$ i obrnuto.*

## Napomena

Gornja definicija podrazumeva da su problemi $\mathcal{P}_1$ i $\mathcal{P}_2$ definisani nad istim nizom promenljivih. Nešto ,,relaksiranija" definicija koja omogućava razmatranje ekvivalentnosti problema nad delimično preklapajućim skupovima promenljivih data je u nastavku.

## Definicija 19

*Dva CSP problema $\mathcal{P}_1$ (nad nizom promenljivih $\mathcal{X}_1$) i $\mathcal{P}_2$ (nad nizom promenljivih $\mathcal{X}_2$) su ekvivalentna u odnosu na niz promenljivih $\mathcal{X}$ ($\mathcal{X} \subseteq \mathcal{X}_1$ i $\mathcal{X} \subseteq \mathcal{X}_2$), u oznaci $\mathcal{P}_1 \equiv_{\mathcal{X}} \mathcal{P}_2$, ako za svako rešenje $\mathbf{s}_1$ problema $\mathcal{P}_1$ postoji rešenje $\mathbf{s}_2$ problema $\mathcal{P}_2$ koje se sa $\mathbf{s}_1$ poklapa na svim promenljivama iz $\mathcal{X}$, kao i obratno.*

# Propagacija ograničenja

**Definicija 20**

*Propagacija ograničenja* (engl. *constraint propagation*) *podrazumeva primenu rezonovanja o ograničenjima u cilju transformacije problema $\mathcal{P}$ u njemu ekvivalentan problem $\mathcal{P}'$ koji je, u nekom smislu, jednostavniji za dalje rešavanje.*

**Šta može uključivati propagacija ograničenja?**

Ovo može uključivati:

- sužavanje domena promenljivih – filtriranje (engl. domain filtering)
- zamenu postojećih ograničenja jednostavnijim
- dodavanje novih ograničenja koja su posledica postojećih
- brisanje redundantnih ograničenja

**Napomena**

Formalno, propagacije ograničenja ćemo predstavljati pravilima oblika:

$$f: \quad \frac{\mathcal{P}}{\mathcal{P}'}$$

gde je $f$ naziv pravila, a $\mathcal{P}'$ problem dobijen primenom pravila $f$ na problem $\mathcal{P}$. Primenu pravila propagacije ćemo često zapisivati i u funkcionalnom obliku: $\mathcal{P}' = f(\mathcal{P})$. Za probleme $\mathcal{P}$ i $\mathcal{P}'$ se podrazumeva da su ekvivalentni u odnosu na najveći zajednički podskup promenljivih.

# Transformacije problema

## Oznake transformacija

Za dati problem $\mathcal{P}$ uvodimo sledeće oznake:

- $\mathcal{P}[D(x) := D]$ označava problem koji nastaje od problema $\mathcal{P}$ sužavanjem domena promenljive $x$ na $D \subset D(x)$
- $\mathcal{P}[C := C']$ označava problem koji nastaje od problema $\mathcal{P}$ zamenom ograničenja $C$ ograničenjem $C'$
- $\mathcal{P}[+C]$ označava problem koji nastaje od problema $\mathcal{P}$ dodavanjem novog ograničenja $C$
- $\mathcal{P}[-C]$ označava problem koji nastaje od problema $\mathcal{P}$ uklanjanjem ograničenja $C$

# Propagacija ograničenja

## Zašto ,,propagacija"?

- Rezonovanje o ograničenjima se obično sprovodi lokalno:
  - najčešće se razmatraju pojedinačna ograničenja
  - ređe, mogu se posmatrati parovi ograničenja ili više ograničenja odjednom
- Informacija o transformaciji problema (npr. sužavanju domena neke promenljive) koja je izvedena na osnovu nekog ograničenja mora se preneti (propagirati) na ostala ograničenja:
  - drugim rečima, mora se ponovo rezonovati o ostalim ograničenjima uzimajući u obzir promenjeni kontekst
- Procedure koje rezonuju o ograničenjima se nazivaju propagatori
  - specijalno, propagatori koji se zasnivaju na sužavanju domena promenljivih nazivaju se algoritmi filtriranja
- Pravila propagacije se primenjuju dokle god je moguće dalje pojednostavljivati problem, tj. dok se ne postigne fiksna tačka

# Sužavanje domena

## Definicija 21

*Neka su data dva ekvivalentna problema $\mathcal{P}$ i $\mathcal{P}'$ nad istim nizom promenljivih $\mathcal{X}$. Tada kažemo da važi $\mathcal{P}' \preceq \mathcal{P}$ ako za svaku promenljivu $x \in \mathcal{X}$ važi $D'(x) \subseteq D(x)$, gde je $D'(x)$ domen promenljive $x$ u $\mathcal{P}'$, a $D(x)$ domen promenljive $x$ u $\mathcal{P}$. Pritom, ako za bar jedno $x$ važi $D'(x) \subsetneq D(x)$, tada važi stroga relacija $\mathcal{P}' \prec \mathcal{P}$.*

## Napomena

Relacija $\mathcal{P}' \prec \mathcal{P}$ znači da je problem $\mathcal{P}'$ nastao sužavanjem domena od problema $\mathcal{P}$.

# Primer – propagacija ograničenja

## Primer 27

*Neka je dat problem $\mathcal{P}$ nad promenljivama $x, y, z$, gde je $D(x) = D(y) = D(z) = \{1, 2, 3\}$, pri čemu su zadata ograničenja: $x < y$ i $y < z$. Iz ograničenja $y < z$ možemo zaključiti da u proizvoljnom rešenju $y$ ne može uzeti vrednost 3. Otuda možemo primeniti pravilo:*

$$UB^{x_1<x_2}: \quad \frac{\mathcal{P}}{\mathcal{P}[D(x_1) := \{v \in D(x_1) \mid v < max(D(x_2))\}]}$$

*za $x_1 := y$ i $x_2 := z$, čime se iz domena promenljive $y$ uklanja vrednost 3, zadržavajući ekvivalentnost. Na sličan način, na osnovu ograničenja $x < y$ možemo iz domena promenljive $y$ ukloniti vrednost 1, primenom pravila:*

$$LB^{x_1<x_2}: \quad \frac{\mathcal{P}}{\mathcal{P}[D(x_2) := \{v \in D(x_2) \mid v > min(D(x_1))\}]}$$

*za $x_1 := x$ i $x_2 := y$.*

# Implicitno ažuriranje ograničenja

## Podsetnik

Ograničenje $C$ nad promenljivama $(x_{i_1}, \ldots, x_{i_r})$ je, po definiciji, uvek podskup skupa $D(x_{i_1}) \times \ldots \times D(x_{i_r})$:

- Otuda, kad god izvršimo transformaciju sužavanja domena $\mathcal{P}[D(x_{i_k}) := D]$ za neku od promenljivih $x_{i_k} \in \mathcal{X}(C)$, potrebno je ažurirati i ograničenje $C$, da bi relacija $C \subseteq D(x_{i_1}) \times \ldots \times D(x_{i_r})$ ostala na snazi
- Smatraćemo da se ovo dešava automatski, tj. da notacija $\mathcal{P}[D(x_{i_k}) := D]$ podrazumeva problem u kome su ažurirana i sva ograničenja nad $x_{i_k}$ tako da ne sadrže $r$-torke koje uključuju obrisane vrednosti iz domena $D(x_{i_k})$

## Primer (nastavak)

*U prethodnom primeru, ograničenje* $x < y$ *se sastojalo iz parova* $\{(1,2),(1,3),(2,3)\}$. *Nakon brisanja vrednosti* 3 *iz domena promenljive* $y$, *ovo ograničenje postaje* $\{(1,2)\}$. *Slično, ograničenje* $y < z$ *koje se sastojalo iz parova* $\{(1,2),(1,3),(2,3)\}$ *nakon brisanja vrednosti* 1 *iz domena promenljive* $y$, *postaje* $\{(2,3)\}$.

# Primer – propagacija ograničenja

## Primer (nastavak)

*Dalje iz $x < y$ i $D(y) = \{2\}$ sledi da mora biti $D(x) = \{1\}$ (primenom pravila $UB^{x<y}$). Na sličan način, iz $y < z$ zaključujemo da mora biti $D(z) = \{3\}$, na osnovu pravila $LB^{y<z}$. Dobijeni problem je atomičan: $D(x) = \{1\}$, $D(y) = \{2\}$ i $D(z) = \{3\}$, pri čemu dodela $\{x = 1, y = 2, z = 3\}$ zadovoljava sva njegova ograničenja, pa je problem rešen.*

# Primer – propagacija ograničenja

## Primer 28

*Neka je ponovo dat isti problem $\mathcal{P}$ nad promenljivama $x, y, z$, sa domenima $D(x) = D(y) = D(z) = \{1, 2, 3\}$ i ograničenjima $x < y$ i $y < z$ kao i malopre. Možemo da zaključimo da je $x < z$, pa možemo dodati novo ograničenje, zadržavajući ekvivalentnost, primenom sledećeg pravila:*

$$TR^{x_1<x_2, x_2<x_3} : \quad \frac{\mathcal{P}}{\mathcal{P}[+C_{x_1<x_3}]}$$

*gde je $C_{x_1<x_3} = \{(u, v) \in D(x_1) \times D(x_3) \mid x_1 < x_3\}$. U ovom slučaju se pravilo primenjuje za $x_1 := x$, $x_2 := y$, $x_3 := z$.*

# Primer – propagacija ograničenja

## Primer 29

*Neka je dat problem nad promenljivama* $x, y, z$*, gde su domeni* $D(x) = D(y) = D(z) = \{1, 2, 3, 4, 5\}$*, a ograničenja su* $x < z$*,* $x < y$ *i* $y < z$*. Ograničenje* $x < z$ *sadrži parove* $(1, 2), (1, 3), (2, 3), \ldots$*. Međutim, iz* $x < y$ *i* $y < z$ *zaključujemo da između* $x$ *i* $z$ *mora da bude vrednost* $y$*, pa se ograničenje* $x < z$ *može redukovati primenom pravila:*

$$BT^{x_1<x_2, x_2<x_3, x_1<x_3} : \quad \frac{\mathcal{P}}{\mathcal{P}[C_{x_1<x_3} := C_{x_1<x_3} \Cap x_2]}$$

*gde je*
$C_{x_1<x_3} \Cap x_2 = \{(u, v) \in D(x_1) \times D(x_3) \mid \exists w \in D(x_2).u < w \wedge w < v\}$*, pri čemu se pravilo primenjuje za* $x_1 := x$*,* $x_2 := y$ *i* $x_3 := z$*.*

# Nepotpunost propagacije

## VAŽNA NAPOMENA

Propagacija kao tehnika rešavanja CSP problema nije dovoljna sama za sebe, jer u opštem slučaju ne svodi CSP problem na rešen ili neuspešan problem.

# Unija problema

## Definicija 22

*Problem $\mathcal{P}$ je ekvivalentan sa unijom problema $\mathcal{P}_1, \ldots, \mathcal{P}_m$, u oznaci $\mathcal{P} \equiv \mathcal{P}_1 \cup \ldots \cup \mathcal{P}_m$, ako, pod pretpostavkom da je sa $\mathcal{X}$ (odnosno $\mathcal{X}_i$) označen niz varijabli problema $\mathcal{P}$ (odnosno $\mathcal{P}_i$), važi:*

- $\mathcal{X} \subseteq \mathcal{X}_i$ $(i \in \{1, \ldots, m\})$
- *za svako rešenje* $\mathbf{s}$ *problema* $\mathcal{P}$ *postoji rešenje* $\mathbf{s}_i$ *nekog od problema* $\mathcal{P}_i$ *koje se sa* $\mathbf{s}$ *poklapa na* $\mathcal{X}$
- *za svako* $i \in \{1, \ldots, m\}$, *ako je* $\mathbf{s}_i$ *proizvoljno rešenje problema* $\mathcal{P}_i$, *tada je restrikcija rešenja* $\mathbf{s}_i$ *na* $\mathcal{X}$ *rešenje problema* $\mathcal{P}$

# Pretraga

## Definicija 23

*Pretraga* *podrazumeva razlaganje CSP problema $\mathcal{P}$ na niz podproblema $\mathcal{P}_1, \ldots, \mathcal{P}_m$ takvih da važi $\mathcal{P} \equiv \mathcal{P}_1 \cup \ldots \cup \mathcal{P}_m$ koji se dalje pojedinačno rešavaju.*

## Kako obično vršimo podelu na podprobleme?

- Najčešće se vrši podela domena $D = D(x)$ neke promenljive $x$ na disjunktne podskupove: $D = D_1 \cup \ldots \cup D_m$
- Ređe, možemo podeliti neko ograničenje $C$ na disjunktne podskupove: $C = C_1 \cup \ldots \cup C_m$

## Napomena

Formalno, pretragu ćemo predstavljati pravilima oblika:

$$f: \quad \frac{\mathcal{P}}{\mathcal{P}_1 \mid \ldots \mid \mathcal{P}_m}$$

gde je $f$ naziv pravila, a $\mathcal{P}_1, \ldots, \mathcal{P}_m$ su podproblemi na koje delimo problem $\mathcal{P}$. Pretragu ćemo takođe zapisivati i u funkcionalnom obliku: $\mathcal{P}_1, \ldots, \mathcal{P}_m = f(\mathcal{P})$.

# Podela domena

## Podela domena

- Podela problema prilikom pretrage se najčešće vrši podelom domena
- Postoje tri osnovne varijante podele domena:
  - enumeracija: isprobavanje jedne po jedne vrednosti iz domena (stablo pretrage je binarno)
  - označavanje: razmatranje zasebnog problema za svaku vrednost iz domena ($n$-arno stablo pretrage)
  - bisekcija: podela domena na dve jednake polovine

# Enumeracija

## Definicija 24

*Pravilo enumeracije:*

$$E^{x,v}: \quad \frac{\mathcal{P}}{\mathcal{P}[D(x) := \{v\}] \mid \mathcal{P}[D(x) := D(x) \setminus \{v\}]}$$

*gde je* $v \in D(x)$, *i* $D(x) \setminus \{v\} \neq \varnothing$.

# Označavanje

## Definicija 25

*Pravilo označavanja:*

$$L^x: \quad \frac{\mathcal{P}}{\mathcal{P}[D(x) := \{v_1\}] \mid \ldots \mid P[D(x) := \{v_k\}]}$$

*gde je* $D(x) = \{v_1, \ldots, v_k\}$ *i* $k \geq 2$.

# Bisekcija

## Definicija 26

*Pravilo bisekcije:*

$$B^x: \quad \frac{\mathcal{P}}{\mathcal{P}[D(x) := \{v \in D(x) \mid v \leq s\}] \mid \mathcal{P}[D(x) := \{v \in D(x) \mid v > s\}]}$$

*gde je $s$ medijalna vrednost uređenog domena $D(x)$ i $|D(x)| \geq 2$.*

# Primer – pretraga

## Primer 30

*Razmotrimo CSP problem $\mathcal{P}$ nad promenljivama $x, y, z$, gde je $D(x) = \{1, 2\}$, $D(y) = \{2, 3\}$ i $D(z) = \{3, 4\}$, pri čemu imamo ograničenja $x < y$ i $y < z$. Primenom pravila označavanja $L^y$, dobijamo dva podproblema:*

$$\mathcal{P}_1 = \mathcal{P}[D(y) := \{2\}]$$

*i*

$$\mathcal{P}_2 = \mathcal{P}[D(y) := \{3\}]$$

*Isti rezultat bismo dobili i primenom pravila enumeracije $E^{y,2}$, kao i pravilom bisekcije $B^y$. Na dobijene podprobleme dalje možemo primeniti propagaciju, kao u prethodnim primerima.*

# Primer – pretraga

## Primer 31

*Neka je dat CSP problem* $\mathcal{P}$ *nad promenljivama* $x, y, z$*, gde je* $D(x) = D(y) = \{1, 2, 3\}$*,* $D(z) = \{1, 2, 3, 4, 5\}$*, a ograničenje je* $\texttt{alldiff}(x, y, z)$*. Primenom pravila enumeracije* $E^{x,1}$ *dobijamo dva podproblema:* $\mathcal{P}_1 = \mathcal{P}[D(x) := \{1\}]$ *i* $\mathcal{P}_2 = \mathcal{P}[D(x) = \{2, 3\}]$*. Daljom primenom pravila enumeracije* $E^{x,2}$ *na* $\mathcal{P}_2$ *dobijamo podprobleme* $\mathcal{P}_2' = \mathcal{P}_2[D(x) := \{2\}]$ *i* $\mathcal{P}_2'' = \mathcal{P}_2[D(x) := \{3\}]$*.*

*Sa druge strane, na problem* $\mathcal{P}$ *smo mogli primeniti i pravilo označavanja* $L^x$*, čime bismo dobili podprobleme* $\mathcal{P}_1 = \mathcal{P}[D(x) := \{1\}]$*,* $\mathcal{P}_2 = \mathcal{P}[D(x) := \{2\}]$ *i* $\mathcal{P}_3 = \mathcal{P}[D(x) := \{3\}]$*.*

*Najzad, primenom pravila bisekcije* $B^z$ *dobili bismo dva podproblema:* $\mathcal{P}_1 = \mathcal{P}[D(z) := \{1, 2, 3\}]$ *i* $\mathcal{P}_2 = \mathcal{P}[D(z) = \{4, 5\}]$*.*

# Pretraga – zaustavljanje

## Primedba

Ako je problem $\mathcal{P}'$ bilo koji od podproblema dobijenih primenom pravila enumeracije, označavanja ili bisekcije na $\mathcal{P}$, tada važi $\mathcal{P}' \prec \mathcal{P}$:

- odavde sledi da se podela na podprobleme ne može primenjivati u nedogled, jer je relacija $\prec$ dobro zasnovana
- sa druge strane, ova pravila se uvek mogu primeniti na problem koji nije atomičan

## Posledica

Uzastopnom primenom pravila enumeracije, označavanja ili bisekcije na proizvoljni CSP problem $\mathcal{P}$, u konačnom broju koraka dolazi se do atomičnih problema. Otuda se pretraga zasnovana na ovim pravilima garantovano zaustavlja, a skup rešenja problema $\mathcal{P}$ je jednak uniji skupova problema svih atomičnih problema dostignutih tokom pretrage.

## Napomena

Za razliku od propagacije, pretraga je kao tehnika sama za sebe dovoljna za rešavanje bilo kog CSP problema. Propagacijom se, sa druge strane, smanjuje prostor pretrage i povećava efikasnost rešavanja.

# Opšti algoritam

## Opšti algoritam

```
function solve (P: CSP problem): set of solutions
begin
    P_prop = propagate_constraints(P)
        if solved(P_prop) then
            if single_solution_mode then
                finished = true
            return solutions(P_prop)
        else if failed(P_prop) then
            return {}
        P_1, ..., P_m = split(P_prop)
        sols = {}
        forall i ∈ {1, ..., m}
            if !finished then
                sols = sols ∪ solve(P_i)
    return sols
end
```

# Opšti algoritam (2)

**propagate_constraints()**

Ova funkcija vrši propagaciju ograničenja, tj. transformiše problem u njemu ekvivalentan, jednostavniji problem. Obično se propagacije ograničenja primenjuju iscrpno, do postizanja tzv. fiksne tačke.

**solved()**

Ova funkcija ispituje da li je problem u rešenoj formi. Način ove provere može zavisiti od prirode samog problema, ali u opštem slučaju se ova provera svodi na ispitivanje atomičnosti problema.

**solutions()**

Ova funkcija za problem u rešenoj formi vraća skup svih njegovih rešenja (ovo je, po pretpostavci, jednostavno uraditi).

**failed()**

Ova funkcija proverava da li je problem neuspešan. To znači da se proverava da li je došlo do pražnjenja domena neke promenljive (engl. domain wipeout), ili do pražnjenja nekog ograničenja.

**split()**

Ova funkcija vrši podelu problema na podprobleme.

**finished**

Globalna promenljiva koja se postavlja na `true` kada je potrebno prekinuti dalju pretragu (npr. u slučaju da se traži samo jedno rešenje koje je već pronađeno).

# Heuristike grananja

## Koje pravilo pretrage primeniti?

- od izbora pravila pretrage u velikoj meri zavisi efikasnost
- na žalost, ne postoji deterministički odgovor na pitanje kako vršiti grananje
- zbog toga je neophodno primeniti različite heuristike
- izbor pravila grananja se sastoji iz dve faze:
  - izbor promenljive po kojoj se vrši grananje
  - izbor vrednosti iz domena koja se prva dodeljuje promenljivoj

## Izbor promenljive za grananje

- promenljiva sa najmanjim domenom
- promenljiva koja učestvuje u najviše ograničenja
- promenljiva koja je ,,najaktivnija" prilikom propagacije

## Izbor vrednosti iz domena

- izbor najmanje vrednosti
- izbor slučajne vrednosti
- izbor medijalne vrednosti
- izbor prethodno dodeljene vrednosti

# Lokalna i globalna konzistentnost

## Podsetnik

- Cilj propagacije ograničenja je da se upotrebom rezonovanja suzi prostor pretrage
- Ovo se obično radi tako što brišemo vrednosti iz domena promenljivih za koje smo utvrdili da nisu deo ni jednog rešenja
  - na ovaj način možemo garantovati da će dobijeni problem biti ekvivalentan polaznom problemu

## Definicija 27

*Vrednost $d \in D(x)$ je (globalno) konzistentna ako postoji rešenje problema u kome je $x = d$. U suprotnom, vrednost $d$ je nekonzistentna.*

## Problem?

- Ispostavlja se da je pronaći vrednosti promenljivih koje jesu konzistentne teško samo po sebi
- Obrnuto – prepoznati *neke* vrednosti koje nisu konzistentne može biti znatno lakše

## Definicija 28

*Vrednost $d \in D(x)$ je lokalno konzistentna u odnosu na ograničenje $C$ ako postoji dodela vrednosti promenljivama koja zadovoljava $C$ a u kojoj je $x = d$. U suprotnom, vrednost $d$ je lokalno nekonzistentna.*

## Primedba

- Vrednost koja je globalno konzistentna je i lokalno konzistentna za svako ograničenje $C$
- Obrnuto, na žalost, ne važi
- Ipak, važi da je lokalno nekonzistentna vrednost i globalno nekonzistentna

# Lokalna konzistentnost

## Dakle, šta je cilj?

- Ideja je da lokalno, na nivou pojedinačnih ograničenja otkrivamo nekonzistentne vrednosti
- Lokalno nekonzistentne vrednosti se brišu iz domena promenljivih, a zatim se ta informacija prenosi ostalim ograničenjima (propagacija)

## Vidovi lokalne konzistentnosti

- Idealno, želimo da pronađemo i uklonimo sve lokalno nekonzistentne vrednosti
  - na ovaj način bismo postigli potpunu lokalnu konzistentnost tog ograničenja
  - u praksi, u zavisnosti od vrste ograničenja, to ponekad može biti preskupo
- U cilju efikasnosti, razmatra se više različitih vidova lokalne konzistentnosti koje nam mogu biti cilj:
  - konzistentnost čvorova, konzistentnost lukova, konzistentnost hiperlukova, konzistentnost granica i td.

# Graf ograničenja

## Definicija 29

*Binarni CSP je CSP koji sadrži isključivo ograničenja arnosti najviše 2.*

## Napomena

Može se pokazati da se svaki CSP može transformisati u njemu ekvivalentan binarni CSP

- ovaj rezultat je od izuzetnog teorijskog značaja, jer se binarni CSP lakše proučava
- ipak, binarizacija CSP-a se u praksi obično ne radi, jer ne doprinosi efikasnosti rešavanja

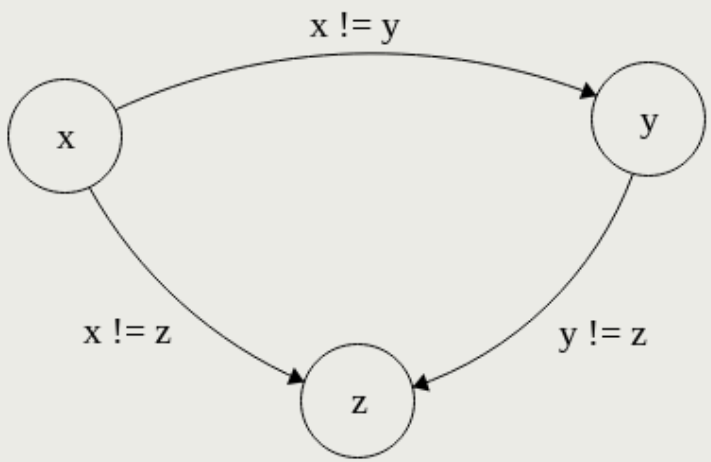
## Definicija 30

*Graf binarnog CSP-a $\mathcal{P}$ je usmereni graf čiji su čvorovi promenljive problema $\mathcal{P}$, a lukovi odgovaraju binarnim ograničenjima.*

# Primer – graf ograničenja

## Primer 32

*Neka je dat CSP problem $\mathcal{P}$ nad promenljivama $x, y, z$ čiji su domeni $D(x) = D(y) = D(z) = \{1, 2\}$, a ograničenja su $x \neq y$, $x \neq z$, $y \neq z$. Graf ovog ograničenja je:*




*NAPOMENA: Iako je graf formalno usmeren, u praksi se obično razmatra njegova neusmerena varijanta.*

# Konzistentnost čvorova

## Definicija 31

*Za CSP $\mathcal{P}$ kažemo da ima svojstvo konzistentnosti čvorova (engl. node consistency), ako za svaku promenljivu $x$, svaka vrednost $d \in D(x)$ zadovoljava sva unarna ograničenja nad promenljivom $x$.*

## Uspostavljanje konzistentnosti čvorova

Konzistentnost lukova problema $\mathcal{P}$ može se uspostaviti iscrpnom primenom pravila oblika:

$$\text{NC-rule} \quad \frac{\mathcal{P}}{\mathcal{P}[D(x) := D(x) \cap C_x]}$$

gde je $C_x$ unarno ograničenje defisano nad promenljivom $x$.

## Algoritam

```
function NC(P: CSP problem) : CSP problem
begin
   foreach variable x
      foreach unary constraint C over x
         P = NC-rule(P, x, C)
   return P
end
```

# Konzistentnost lukova

**Definicija 32**

*Za binarni CSP kažemo da ima svojstvo konzistentnosti lukova (engl. arc consistency), ako za svako binarno ograničenje $C$ nad promenljivama $(x, y)$ važi:*

- *za svaku vrednost $a \in D(x)$ postoji vrednost $b$ iz domena $D(y)$ takva da $(a, b) \in C$*
- *za svaku vrednost $b \in D(y)$ postoji vrednost $a$ iz domena $D(x)$ takva da $(a, b) \in C$*

**Napomena**

Vrednost $b$ (odnosno $a$) naziva se podrška (engl. support) za $a$ (odnosno $b$) u odnosu na ograničenje $C$. CSP ima svojstvo konzistentnosti lukova ako u svakom binarnom ograničenju svaka vrednost iz domena jedne promenljive ima podršku u odnosu na to ograničenje u domenu druge promenljive.

**Napomena**

Konzistentnost lukova je najjače svojstvo od svih lokalnih svojstava konzistentnosti za binarna ograničenja. Ona podrazumeva uklanjanje svih lokalno nekonzistentnih vrednosti.

# Konzistentnost lukova

## Uspostavljanje konzistentnosti lukova

Konzistentnost lukova binarnog CSP-a $\mathcal{P}$ se može uspostaviti iscrpnom primenom sledećih pravila:

$$\text{AC-rule1:} \quad \frac{\mathcal{P}}{\mathcal{P}[D(x) := D(x) \cap C_{x,y} \downarrow x]}$$

gde je $C_{x,y}$ ograničenje nad $(x, y)$, a $C_{x,y} \downarrow x = \{a \mid (a, b) \in C_{x,y}\}$ je projekcija ograničenja $C_{x,y}$ na $x$, i:

$$\text{AC-rule2:} \quad \frac{\mathcal{P}}{\mathcal{P}[D(y) := D(y) \cap C_{x,y} \downarrow y]}$$

gde je $C_{x,y}$ ograničenje nad $(x, y)$, a $C_{x,y} \downarrow y = \{b \mid (a, b) \in C_{x,y}\}$ projekcija ograničenja $C_{x,y}$ na $y$.

# Algoritam AC3

## Algoritam

```
function AC3(P: CSP problem): CSP problem
begin
   foreach binary constraint C ∈ P over (x, y)
      push pairs (C, x) and (C, y) to queue Q
   while Q not empty
      pop a pair (C_{x,y}, z) from queue Q
      if z = x then
         P = AC-rule1(P, x, y, C_{x,y})
         if D(x) changed
            foreach bin. cons. C' ∈ P \ {C_{x,y}} over (x, y') or (y', x)
               push (C', y') to Q (if not already in Q)
      else // z = y
         P = AC-rule2(P, x, y, C_{x,y})
         if D(y) changed
            foreach bin. cons. C' ∈ P \ {C_{x,y}} over (y, x') or (x', y)
               push (C', x') to Q (if not already in Q)
return P
end
```

# Opšti algoritam filtriranja za binarna ograničenja

## Algoritam

```
function AC-rule1(P: CSP problem, x: var, y: var, C: cons): CSP problem
begin
   foreach v ∈ D(x)
      supported = false
      foreach w ∈ D(y)
         if (v, w) ∈ C then
            supported = true
            break
      if supported = false then
         D(x) = D(x) \ {v}
   return P
end
function AC-rule2(P: CSP problem, x: var, y: var, C: cons): CSP problem
begin
   foreach w ∈ D(y)
      supported = false
      foreach v ∈ D(x)
         if (v, w) ∈ C then
            supported = true
            break
      if supported = false then
         D(y) = D(y) \ {w}
   return P
end
```

# Složenost AC3 algoritma

## Složenost filtriranja domena

- za svaku vrednost iz $D(x)$ proveravamo da li ima podršku u $D(y)$
- složenost u najgorem slučaju: $O(d_x d_y)$, gde je $d_x = |D(x)|$ a $d_y = |D(y)|$
- ako je ograničenje dato nekom specifičnom relacijom, mogu postojati efikasniji algoritmi

## Složenost kompletnog algoritma

- filtriranje domena se poziva onoliko puta koliko puta se ograničenja dodaju u $Q$
- neka je $m$ broj binarnih ograničenja
- inicijalno, svako ograničenje se dodaje dva puta
- ograničenje $C_{x,y}$ se može ponovo dodati u $Q$, ali najviše $d_x + d_y$ puta
  - ovo je zato što se ograničenje ponovo dodaje u $Q$ samo kada se obriše neka od vrednosti iz domena njegovih promenljivih
- ako sa $d$ označimo veličinu najvećeg domena, tada je broj dodavanja ograničenja u $Q$ najviše $2m + 2md$
- otuda je složenost algoritma $O((2m + 2md) \cdot d^2) = O(md^3)$

# Analiza AC3 algoritma

## Da li se AC3 može dodatno optimizovati?

- Algoritam AC3 uvek vrši proveru svih vrednosti u domenu promenljive $y$, da bi utvrdio da li i dalje imaju podršku u domenu $x$ koji je promenjen
  - najčešće su samo neke od njih izgubile podršku, a za ostale je to nepotreban posao
- Ovo je zato što se nigde ne vodi evidencija o podrškama za pojedinačne vrednosti
- Alternativa: proveravati samo one vrednosti za $y$ kojima je obrisana vrednost za $x$ bila podrška
  - ovo zahteva čuvanje mnogo informacija
- Za ovakve algoritme kažemo da su fine granulacije (engl. fine-grained), za razliku od AC3 koji je grube granulacije (engl. coarse-grained)

# Algoritam AC4

## Algoritam

```
function AC4(P: CSP problem): CSP problem
begin
   /* init */
   foreach bin. cons. C // over (x, y)
      foreach v ∈ D(x)
         counter[x, v, C] = |{w ∈ D(y) | (v, w) ∈ C}|
         if counter[x, v, C] = 0 then D(x) = D(x) \ {v}, push (x, v) to Q
         add (x, v, C) to list S[y, w] for all w ∈ D(y) s.t. (v, w) ∈ C
      foreach w ∈ D(y)
         counter[y, w, C] = |{v ∈ D(x) | (v, w) ∈ C}|
         if counter[y, w, C] = 0 then D(y) = D(y) \ {w}, push (y, w) to Q
         add (y, w, C) to list S[x, v] for all v ∈ D(x) s.t. (v, w) ∈ C
   /* propagation */
   while Q not empty
      pop (x, v) from Q
      foreach (y, w, C) ∈ S[x, v]
         if w ∈ D(y) then
            counter[y, w, C] = counter[y, w, C] − 1
            if counter[y, w, C] = 0 then
               D(y) = D(y) \ {w}, push (y, w) to Q
   return P
end
```

# Analiza algoritma AC4

## Složenost algoritma

Može se pokazati da je složenost algoritma AC4 $O(md^2)$ u najgorem slučaju. Ispostavlja se da je ovo optimalna složenost za uspostavljanje konzistentnosti lukova.

## Da li to znači da je AC4 bolji od AC3?

- Ispostavi se da ovaj algoritam gotovo uvek dostiže svoj najgori slučaj
- Algoritam sadrži dosta struktura podataka čija je inicijalizacija i održavanje skupo
- U praksi, AC3 algoritam je jednostavniji za implementaciju i radi brže
  - naročito u slučaju kada imamo ograničenja sa specifičnim algoritmima filtriranja
  - algoritam AC4 ne omogućava da se iskoriste ti specifični algoritmi

# Ne-binarni CSP

## Uopštenje na ne-binarni CSP

U slučaju da postoje ograničenja arnosti veće od 2, moramo uopštiti pojam grafa CSP problema:

- ograničenjima veće arnosti odgovaraju hiper-lukovi koji povezuju više čvorova
- grafove koji sadrže hiper-lukove (koje predstavljamo uređenim $r$-torkama) nazivamo hiper-grafovima

## Definicija 33

*Za dati CSP problem $\mathcal{P}$, njegov hiper-graf će za čvorove imati promenljive problema $\mathcal{P}$, a njegovi hiper-lukovi odgovaraće ograničenjima arnosti veće ili jednake 2 u problemu $\mathcal{P}$.*

# Konzistentnost hiper-lukova

## Definicija 34

*Za CSP kažemo da ima svojstvo konzistentnosti hiper-lukova (engl. hyper-arc consistency) ako za svako ograničenje $C$ nad promenljivama $(x_{i_1}, \ldots, x_{i_r})$ $(r \geq 2)$ važi da za svaku promenljivu $x_{i_k}$ i vrednost $d_{i_k} \in D(x_{i_k})$ postoje vrednosti $d_{i_1} \in D(x_{i_1}), \ldots, d_{i_{k-1}} \in D(x_{i_{k-1}}), d_{i_{k+1}} \in D(x_{i_{k+1}}), \ldots, D(x_{i_r})$ takve da je $(d_{i_1}, \ldots, d_{i_{k-1}}, d_{i_k}, d_{i_{k+1}}, \ldots, d_{i_r}) \in C$.*

## Napomena

Konzistentnost hiper-lukova podrazumeva da svaka vrednost iz domena jedne promenljive ima podršku u vidu $r$-torke koja zadovoljava ograničenje, a koja uključuje tu vrednost.

## Primedba

U literaturi se za konzistentnost hiper-lukova koriste i termini uopštena konzistentnost lukova (engl. generalized arc consistency (GAC) i domenska konzistentnost (engl. domain consistency).

# Konzistentnost hiper-lukova

## Uspostavljanje konzistentnosti hiper-lukova

Konzistentnost hiper-lukova CSP-a $\mathcal{P}$ se može uspostaviti iscrpnom primenom sledećih pravila:

$$\text{GAC-rule[k]:} \quad \frac{\mathcal{P}}{\mathcal{P}[D(x_{i_k}) := D(x_{i_k}) \cap C_{x_{i_1},\ldots,x_{i_r}} \downarrow x_{i_k}]}$$

gde je $C_{x_{i_1},\ldots,x_{i_r}}$ ograničenje nad $(x_{i_1},\ldots,x_{i_r})$, a $C_{x_{i_1},\ldots,x_{i_r}} \downarrow x_{i_k} = \{d_{i_k} \mid (d_{i_1},\ldots,d_{i_{k-1}},d_{i_k},d_{i_{k+1}},\ldots,d_{i_r}) \in C_{x_{i_1},\ldots,x_{i_r}}\}$.

# Algoritam GAC3

## Algoritam

```
function GAC3(P: CSP problem): CSP problem
begin
   foreach non-unary constraint C ∈ P // over (x_{i_1}, ..., x_{i_r})
      push pairs (C, x_{i_k}) to queue Q for all k ∈ {1, ..., r}
   while Q not empty
      pop a pair (C_{x_{i_1},...,x_{i_r}}, z) from queue Q
      assume z = x_{i_k}
      P = AC-rule[k](P, x_{i_1}, ..., x_{i_r}, C_{x_{i_1},...,x_{i_r}})
         if D(x_{i_k}) changed
            foreach non-unary cons. C' ∈ P \ {C_{x_{i_1},...,x_{i_r}}} over y_{j_1}, ..., y_{j_l}, x_{i_k}, y_{j_{l+1}}, ..., y_{j_m}
               push (C', y_{j_t}) to Q for all t ∈ {1, 2, ..., m} (if not already in Q)
return P
end
function AC-rule[k](P : CSP problem, x_{i_1}, ..., x_{i_r}: vars, C: constraint) : CSP problem
begin
   foreach v ∈ D(x_{i_k})
      supported = false
      foreach (d_{i_1}, ..., d_{i_{k-1}}, d_{i_{k+1}}, ..., d_{i_r}) ∈ D(x_{i_1}) × ... × D(x_{i_{k-1}}) × D(x_{i_{k+1}}) × ... × D(x_{i_r})
         if (d_{i_1}, ..., d_{i_{k-1}}, v, d_{i_{k+1}}, ..., d_{i_r}) ∈ C then supported = true; break
      if supported = false then D(x_{i_k}) = D(x_{i_k}) \ {v}
   return P
end
```

# Složenost GAC3 algoritma

## Složenost filtriranja domena

- za svaku vrednost iz $D(x_{i_k})$ proveravamo da li ima podršku u ostalim domenima
- složenost u najgorem slučaju: $O(d^r)$, gde je $d$ veličina najvećeg domena
- u slučaju globalnih ograničenja vrlo često postoje efikasniji algoritmi filtiranja
  - vrlo često se mogu istovremeno filtrirati domeni svih promenljivih koje učestvuju u ograničenju

## Složenost kompletnog algoritma

- filtriranje domena se poziva onoliko puta koliko puta se ograničenja dodaju u $Q$
- neka je $m$ broj ne-unarnih ograničenja i neka je maksimalna arnost $r$
- inicijalno, svako ograničenje se dodaje najviše $r$ puta
- ograničenje $C$ arnosti $\leq r$ se može ponovo dodati u $Q$, ali najviše $r \cdot r \cdot d$ puta
- ukupan broj dodavanja ograničenja u $Q$ je, dakle, najviše $mr + mr^2 d$
- otuda je složenost algoritma $O((mr + mr^2 d) \cdot d^r) = O(mr^2 d^{r+1})$

# Konzistentnost hiper-lukova i globalna ograničenja

## Definicija 35

*Globalno ograničenje* C *je ograničenje zadato relacijom sa specifičnom semantikom koje je ekvivalentno konjunkciji više jednostavnijih ograničenja.*

## Šta ovo znači?

- semantika globalnog ograničenja se može simulirati konjunkcijom većeg broja jednostavnijih ograničenja
- ipak, globalno ograničenje omogućava da se kompaktnije izraze neki složeni uslovi
- dodatno, globalna ograničenja često omogućavaju postizanje ,,jačeg" filtriranja

# Teorema o globalnom ograničenju

## Teorema 1

*Neka je $C$ globalno ograničenje u problemu $\mathcal{P}$ koje je ekvivalentno konjunkciji ograničenja $C_1, \ldots, C_k$. Neka je $\mathcal{P}'$ problem koji nastaje uspostavljanjem konzistentnosti hiper-lukova nad $C$, a $\mathcal{P}''$ problem koji nastaje uspostavljanjem konzistentnosti hiper-lukova nad ograničenjima $C_1, \ldots, C_k$. Tada važi $\mathcal{P}' \preceq \mathcal{P}''$.*

## Šta ovo znači

Teorema govori da se uspostavljanjem konzistentnosti hiper-lukova nad globalnim ograničenjem ne može postići ,,slabije" filtriranje domena u odnosu na postizanje iste vrste konzistentnosti nad ekvivalentnom konjunkcijom jednostavnijih ograničenja. U praksi, obično je moguće postići ,,jače" filtriranje upotrebom globalnih ograničenja.

# Teorema o globalnom ograničenju

## Dokaz

*Primetimo da ako neka vrednost $d \in D(x)$ nema podršku u nekom ograničenju $C_i$ $(1 \leq i \leq k)$, tada ga neće imati ni u globalnom ograničenju $C$, jer bi svaka r-torka koja zadovoljava $C$ a sadrži $d$ zadovoljavala i sva ograničenja $C_1, \ldots, C_k$, pa bi $d$ moralo da ima podršku i u tim ograničenjima. Dakle, svaka vrednost $d$ koja je filtrirana uspostavljanjem konzistentnosti hiper-lukova nad $C_1, \ldots, C_k$ biće filtrirana i uspostavljanjem iste konzistentnosti nad globalnim ograničenjem $C$.*

# Primer

## Primer 33

*Razmotrimo ponovo problem $\mathcal{P}$ nad promenljivama $x, y, z$, gde je $D(x) = D(y) = D(z) = \{1, 2\}$, a ograničenja su $x \neq y$, $x \neq z$ i $y \neq z$. Lako se da proveriti da $\mathcal{P}$ ima svojstvo konzistentnosti (hiper-) lukova. Ipak, ovaj problem nije konzistentan.*
*Sa druge strane, ako bismo data ograničenja zamenili jednim ekvivalentnim globalnim ograničenjem* `alldiff`$(x, y, z)$*, tada bi uspostavljanjem konzistentnosti hiper-lukova dobili problem u kom bi domeni sve tri promenljive bili prazni (jer ni jedna vrednost ni u jednom domenu nema podršku u odnosu na ovo globalno ograničenje). Dakle, uspostavljanem konzistentnosti hiper-lukova nad globalnim ograničenjem postižemo „jače" filtriranje domena.*

# Algoritmi filtiranja

## Algoritmi filtriranja za globalna ograničenja

- Postizanje konzistentnosti hiper-lukova nad globalnim ograničenjem zahteva upotrebu specifičnih procedura filtriranja koje su prilagođene semantici globalnog ograničenja
- Za svaki tip globalnog ograničenja razvija se poseban algoritam filtriranja
- Na primer, algoritmi filtriranja za `alldiff` i `gcc` zasnovani su na određivanju maksimalnog protoka kroz mrežu
- Ovi algoritmi mogu biti veoma efikasni i mogu istovremeno filtrirati domene svih promenljivih tog ograničenja

# Odnosi između globalne i konzistentnosti hiper-lukova

## Važi sledeće:

- konzistentnost lukova ne povlači konzistentnost problema
- konzistentnost hiper-lukova ne povlači konzistentnost problema
- konzistentnost problema ne povlači konzistentnost lukova (pa ni hiper-lukova)

Prva stavka sledi iz prethodnog primera. Za druge dve stavke slede primeri u nastavku.

# Primer

## Primer 34

*Neka je dat problem $\mathcal{P}$ nad promenljivama $x, y, z, u$, sa domenom $D(x) = D(y) = D(z) = D(u) = \{1, 2, 3\}$ i ograničenjima* $\texttt{alldiff}(x, y, z)$, $\texttt{alldiff}(y, z, u)$ *i* $\texttt{alldiff}(x, y, u)$. *Svaka od vrednosti u domenima promenljivih ima podršku u svakom od ograničenja, pa problem ima svojstvo konzistentnosti hiper-lukova. Ipak, ovaj problem je nekonzistentan. Naime, uzmimo bilo koju dodelu vrednosti koja zadovoljava ograničenje* $\texttt{alldiff}(x, y, z)$, *na primer,* $x = 1, y = 2, z = 3$. *Iz* $\texttt{alldiff}(x, y, u)$ *i* $x = 1, y = 2$, *sledi da mora biti* $u = 3$. *Otuda ne može važiti* $\texttt{alldiff}(y, z, u)$. *Slično je u svim ostalim slučajevima.*

# Primer

## Primer 35

*Neka je dat problem $\mathcal{P}$ nad promenljivama $x, y, z, u$, gde je $D(x) = D(y) = D(z) = \{1, 2, 3\}$ i $D(u) = \{3, 4, 5\}$, pri čemu imamo ograničenje* $\texttt{alldiff}(x, y, z, u)$. *Ovaj problem je konzistentan (npr. jedno rešenje je $x = 1, y = 2, z = 3, u = 4$). Ipak, ovaj problem nema svojstvo konzistentnosti hiper-lukova, jer npr. vrednost $u = 3$ nema podršku u ograničenju* $\texttt{alldiff}(x, y, z, u)$.

# Rešavanje optimizacionih problema

## Kako se rešavaju optimizacioni problemi?

Pretpostavimo da je cilj maksimizacija funkcije $f(x_1, \ldots, x_n)$. Jedna moguća strategija pronalaženja optimalnog rešenja je tzv. linearna strategija:

- Kada se pronađe neko rešenje $(d_1, \ldots, d_n)$, određuje se vrednost funkcije $d = f(d_1, \ldots, d_n)$
- Dodajemo ograničenje $f(x_1, \ldots, x_n) > d$ u skup ograničenja i nastavljamo sa pretragom
  - zbog ovog dodatnog ograničenja, u obzir dolaze samo rešenja koja daju veću vrednost funkcije $f$
- Ako drugih rešenja nema, tada je pronađeno rešenje optimalno
- U suprotnom, ako pronađemo novo rešenje $(d'_1, \ldots, d'_n)$ za koje je $f(d'_1, \ldots, d'_n) = d'$, tada je, zbog ranije dodatog ograničenja, sigurno $d' > d$, pa pamtimo $(d'_1, \ldots, d'_n)$ kao optimalno rešenje i uvodimo novo ograničenje $f(x_1, \ldots, x_n) > d'$, a zatim nastavljamo pretragu, i td.

# Pregled

# Primer – Polica sa knjigama

## Primer 36

*Data je polica za knjige dužine L, kao i skup od n knjiga različite debljine (označimo sa $D_i$ debljinu i-te knjige). Da li postoji podskup datog skupa knjiga tako da može idealno popuniti policu, tj. da je ukupna debljina izabranih knjiga jednaka L?*

# Primer – Selidba

## Primer 37

*Prilikom selidbe, stvari se pakuju u torbe jednakog kapaciteta $S$. Svaka stvar ima svoju veličinu, a prilikom pakovanja ne sme se prekoračiti kapacitet torbe. Potrebno je spakovati stvari u torbe tako da broj potrebnih torbi bude minimalan.*

# Primer – Sečenje tepiha

## Primer 38

*Data je rolna tepiha širine $W$. Potrebno je iseći određeni broj tepiha pravougaonog oblika unapred zadatih dimenzija (neka su $w_i$ i $l_i$, respektivno, širina i dužina i-tog tepiha). Sve dimenzije su celi brojevi zadati u centimetrima. Pretpostavka je da je dozvoljeno sečenje samo ,,uzduž" ili ,,popreko", ali ne i dijagonalno (tj. sve ivice pravougaonika moraju biti paralelne sa ivicama same rolne). Takođe, pretpostavljamo da je za svaki tepih bar jedna od njegovih dimenzija $w_i$ ili $l_i$ manja ili jednaka $W$. Odrediti minimalnu dužinu rolne $L$ koja je dovoljna za izradu tepiha datih dimenzija.*

# Primer – Podela novca

## Primer 39

*Dva lopova nakon uspešne krađe žele da podele novac. U džaku sa ukradenim novcem je veliki broj novčanica u različitim apoenima. Da li je moguće da oni podele novac na dva potpuno jednaka dela?*

*NAPOMENA: Ulaz problema je broj novčanica n, kao i niz brojeva dužine n koji predstavljaju vrednosti novčanica (među kojima može biti i istih vrednosti). Na izlazu bi trebalo prikazati niz novčanica prvog i niz nočanica drugog lopova, pod pretpostavkom da odgovarajuća ravnopravna podela postoji.*

# Primer – Raspored ispita

## Primer 40

*Potrebno je organizovati ispitni rok na fakultetu. Imamo m studentskih grupa kao i n različitih ispita. Pritom, svaka grupa polaže tačno određene ispite (ovi uslovi su zadati matricom E logičkih vrednosti, gde $E[i, j] = true$ označava da i-ta grupa polaže j-ti ispit). Svaki ispit se organizuje samo jednom, tj. sve grupe koje polažu neki ispit polažu ga u istom terminu. Zbog ograničenog broja sala, svakoga dana je moguće organizovati najviše r ispita. Ni jedna grupa ne sme imati dva ili više ispita istog dana, niti sme imati ispite više od dva uzastopna dana. Da li je moguće napraviti ovakav raspored, ako se zna da ispitni rok traje N dana.*

*NAPOMENA: Može se razmatrati i optimizaciona varijanta: minimizovati N, tj. obezbediti da ispitni rok traje što kraće.*

# Pregled

# Literatura

## Literatura


- Apt, Krzysztof. Principles of constraint programming. Cambridge university press, 2003.
- Rossi, Francesca, Peter Van Beek, and Toby Walsh, eds. Handbook of constraint programming. Elsevier, 2006.
- Global Constraint Catalog: `https://sofdem.github.io/gccat/gccat/sec5.html`
- Van Hoeve, Willem-Jan. The alldifferent constraint: A survey. arXiv preprint cs/0105015 (2001).
- MiniZinc: `https://www.minizinc.org/`
- MiniZinc Tutorial: `https://www.minizinc.org/doc-2.6.1/en/part_2_tutorial.html`
- MiniZinc Challenge: `https://www.minizinc.org/challenge.html`
- CSPLib (kolekcija CSP problema): `https://www.csplib.org/`