Jazyk na popis stolných hier

Truc Lam, Bui
Školiteľ: RNDr. Jozef Šiška

20. februára 2020

Motivácia

Prečo? Čo je zaujímavé?

- stolné hry nemôžu byť výpočtovo náročné, lebo za sledovanie chodu hry sú zodpovední ľudia (hráči)
- ako zachytiť všetko to, čo sa môže v pravidlách vyskytnúť?
- neušetrí počítačový čas, ale ľudský čas

Ako na to

Koncepty z hier

- akcie hráčov, čo môžu kedy vykonávať
- čo jednotliví hráči vidia
- čo je ich cieľom

Koncepty z programovania

- ako vhodne vyjadriť pravidlá
- čo je špecifické pre doménu stolných hier?

Ako na to

Koncepty z hier

- akcie hráčov, čo môžu kedy vykonávať
- čo jednotliví hráči vidia
- čo je ich cieľom

Koncepty z programovania

- ako vhodne vyjadriť pravidlá
- čo je špecifické pre doménu stolných hier?

Aký jazyk?

Spektrum programovacích jazykov

- na jednom konci: veľmi štruktúrované a precízne, výpočtovo jednoduché na interpretovanie a kompilovanie (C/C++, ...)
  - "umelé" pojmy ako funkcie, procedúry, premenné, ...
  - problém: ako meniť program za behu?
- ...
- na druhom konci: neštruktúrované, len kolekcia faktov a pravidiel, výpočtovo náročné (logika, ...)
  - zmena programu za behu: iba pridáme (odvodíme) nové fakty/pravidlá

Aký jazyk?

Spektrum programovacích jazykov

- na jednom konci: veľmi štruktúrované a precízne, výpočtovo jednoduché na interpretovanie a kompilovanie (C/C++, ...)
  - "umelé" pojmy ako funkcie, procedúry, premenné, ...
  - problém: ako meniť program za behu?
- ...
- na druhom konci: neštruktúrované, len kolekcia faktov a pravidiel, výpočtovo náročné (logika, ...)
  - zmena programu za behu: iba pridáme (odvodíme) nové fakty/pravidlá

Aký jazyk?

Spektrum programovacích jazykov

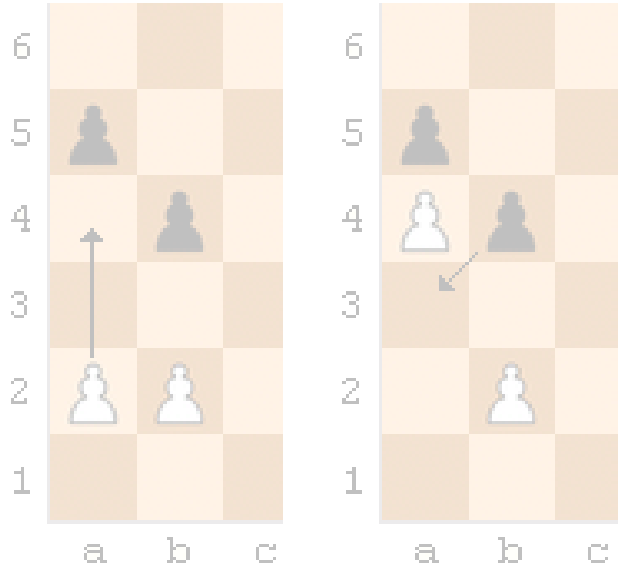
- na jednom konci: veľmi štruktúrované a precízne, výpočtovo jednoduché na interpretovanie a kompilovanie (C/C++, ...)
  - "umelé" pojmy ako funkcie, procedúry, premenné, ...
  - problém: ako meniť program za behu?
- ...
- na druhom konci: neštruktúrované, len kolekcia faktov a pravidiel, výpočtovo náročné (logika, ...)
  - zmena programu za behu: iba pridáme (odvodíme) nové fakty/pravidlá

Aký jazyk?

Stolné hry

Predstavme si, že niekomu vysvetľujeme pravidlá hry.

- časté úpravy už predtým definovaných pojmov, možnosť meniť pravidlá v presne vymedzených prípadoch
- dodatky na riešenie konfliktných situácii a situácii s viacerými interpretáciami

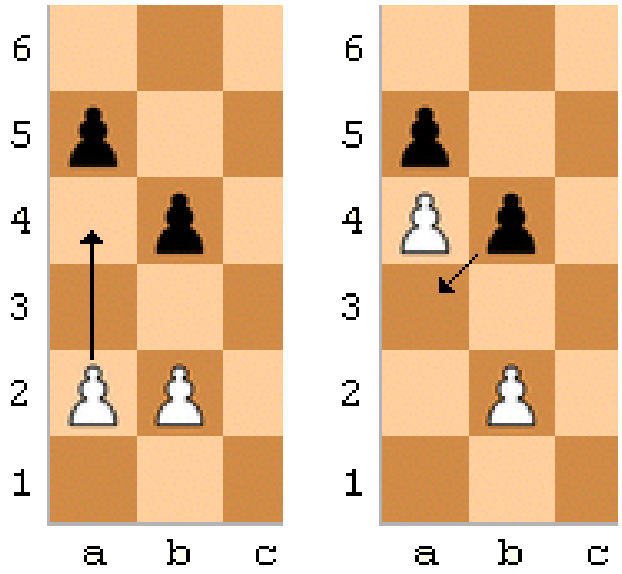


Aký jazyk?

Stolné hry

Predstavme si, že niekomu vysvetľujeme pravidlá hry.

- časté úpravy už predtým definovaných pojmov, možnosť meniť pravidlá v presne vymedzených prípadoch
- dodatky na riešenie konfliktných situácii a situácii s viacerými interpretáciami

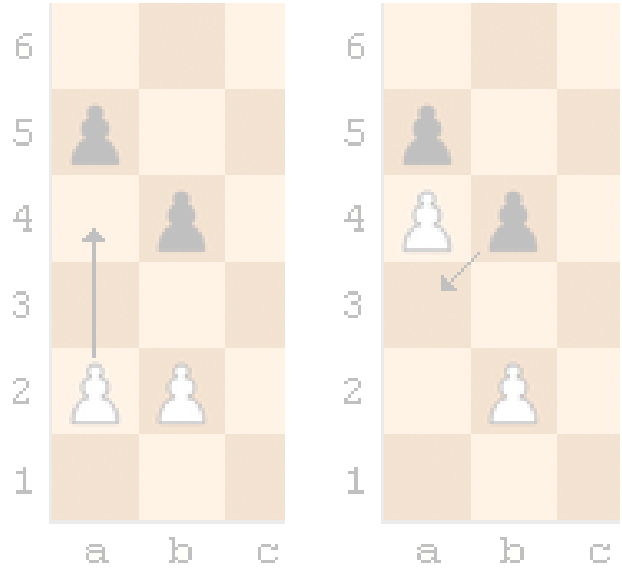


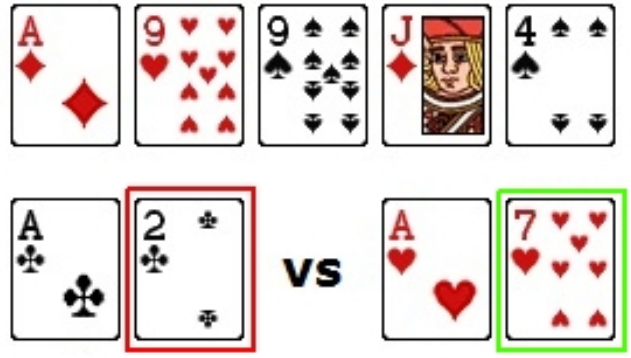
Aký jazyk?

Stolné hry

Predstavme si, že niekomu vysvetľujeme pravidlá hry.

- časté úpravy už predtým definovaných pojmov, možnosť meniť pravidlá v presne vymedzených prípadoch
- dodatky na riešenie konfliktných situácii a situácii s viacerými interpretáciami





Logika

Čo sa tým chápe

- modely: vety chápeme ako tvrdenia o tom, čo má vo svete platiť

$$val(A \Rightarrow B) = \begin{cases} 1, & \text{ak } val(A) = 0 \text{ alebo } val(B) = 1 \\ 0, & \text{inak.} \end{cases}$$

- dôkazy: vety chápeme ako atómy, ktoré môžeme používať v ďalšom odvodzovaní

$$\frac{A \Rightarrow B, \quad A}{B}$$

Logika

Čo sa tým chápe

- modely: vety chápeme ako tvrdenia o tom, čo má vo svete platiť

$$val(A \Rightarrow B) = \begin{cases} 1, & \text{ak } val(A) = 0 \text{ alebo } val(B) = 1 \\ 0, & \text{inak.} \end{cases}$$

- dôkazy: vety chápeme ako atómy, ktoré môžeme používať v ďalšom odvodzovaní

$$\frac{A \Rightarrow B, \quad A}{B}$$

Logika

Ako ju využiť v stolných hrách?

GDL-II

- reprezentácia aktuálneho stavu hry
- v každom časovom okamihu odvodí:
  - *next*: čo sa má udiať (čo má platiť v ďalšom kroku)
  - *legal*: čo môžu jednotliví hráči vykonať
  - *sees*: čo jednotliví hráči uvidia
  - *terminal* a *goal*: či hra skončila a ak áno, aké je výsledné skóre jednotlivých hráčov
- stratifikovaný Datalog

Logika

Ako ju využiť v stolných hrách?

GDL-II

- reprezentácia aktuálneho stavu hry
- v každom časovom okamihu odvodí:
  - *next*: čo sa má udiať (čo má platiť v ďalšom kroku)
  - *legal*: čo môžu jednotliví hráči vykonať
  - *sees*: čo jednotliví hráči uvidia
  - *terminal* a *goal*: či hra skončila a ak áno, aké je výsledné skóre jednotlivých hráčov
- stratifikovaný Datalog

Logika

Ako ju využiť v stolných hrách?

GDL-II

- reprezentácia aktuálneho stavu hry
- v každom časovom okamihu odvodí:
  - *next*: čo sa má udiať (čo má platiť v ďalšom kroku)
  - *legal*: čo môžu jednotliví hráči vykonať
  - *sees*: čo jednotliví hráči uvidia
  - *terminal* a *goal*: či hra skončila a ak áno, aké je výsledné skóre jednotlivých hráčov
- stratifikovaný Datalog

GDL-II: nedostatky

- nedajú sa priamo meniť iné pravidlá; len nepriamo tým, že ovplyvníme niektorý predpoklad

$$\{\text{not } \mathit{rule\_disallowed} \Rightarrow (\ldots),\ a \Rightarrow \mathit{rule\_disallowed}\}$$

- vždy je práve 1 interpretácia toho, čo sa má stať; nedajú sa vyjadriť sporné a nejasné (vágne) pravidlá $\rightarrow$ neprirodzené
- neumožňuje implikáciu v hlave pravidla $(a \Rightarrow (b \Rightarrow c))$
- ...

GDL-II: nedostatky

- nedajú sa priamo meniť iné pravidlá; len nepriamo tým, že ovplyvníme niektorý predpoklad

$$\{\text{not } \mathit{rule\_disallowed} \Rightarrow (\ldots),\ a \Rightarrow \mathit{rule\_disallowed}\}$$

- vždy je práve 1 interpretácia toho, čo sa má stať; nedajú sa vyjadriť sporné a nejasné (vágne) pravidlá $\rightarrow$ neprirodzené
- neumožňuje implikáciu v hlave pravidla $(a \Rightarrow (b \Rightarrow c))$
- ...

GDL-II: nedostatky

- nedajú sa priamo meniť iné pravidlá; len nepriamo tým, že ovplyvníme niektorý predpoklad

$$\{\text{not } \mathit{rule\_disallowed} \Rightarrow (\ldots),\ a \Rightarrow \mathit{rule\_disallowed}\}$$

- vždy je práve 1 interpretácia toho, čo sa má stať; nedajú sa vyjadriť sporné a nejasné (vágne) pravidlá $\rightarrow$ neprirodzené
- neumožňuje implikáciu v hlave pravidla $(a \Rightarrow (b \Rightarrow c))$
- ...

Prototyp

Generický logický systém.

- v každom kroku aplikujeme niektoré odvodzovacie pravidlo; opakujeme, kým sa dá
- iné poradie aplikácii môže viesť do iného stavu; všetky výsledné stavy považujeme za korektné *interpretácie* pravidiel hry
- odvodenie $\bot$ chápeme tak, že sme zle interpretovali pravidlá; stav zahodíme

Prototyp

Generický logický systém.

- v každom kroku aplikujeme niektoré odvodzovacie pravidlo; opakujeme, kým sa dá
- iné poradie aplikácii môže viesť do iného stavu; všetky výsledné stavy považujeme za korektné *interpretácie* pravidiel hry
- odvodenie $\bot$ chápeme tak, že sme zle interpretovali pravidlá; stav zahodíme

Prototyp

Generický logický systém.

- v každom kroku aplikujeme niektoré odvodzovacie pravidlo; opakujeme, kým sa dá
- iné poradie aplikácii môže viesť do iného stavu; všetky výsledné stavy považujeme za korektné *interpretácie* pravidiel hry
- odvodenie $\perp$ chápeme tak, že sme zle interpretovali pravidlá; stav zahodíme

Prototyp

Generický logický systém.

- v každom kroku aplikujeme niektoré odvodzovacie pravidlo; opakujeme, kým sa dá
- iné poradie aplikácii môže viesť do iného stavu; všetky výsledné stavy považujeme za korektné *interpretácie* pravidiel hry
- odvodenie $\perp$ chápeme tak, že sme zle interpretovali pravidlá; stav zahodíme

Prototyp

$$\frac{+X, \quad \text{not } (-X)}{X} \qquad \text{(ztrvácnenie)}$$

$$\frac{X \Rightarrow Y, \quad X}{+Y} \qquad \text{(implikácia)}$$

$$\frac{X \multimap Y, \quad +X, \quad \text{not } (-X)}{-X, \quad +Y} \qquad \text{(lineárna impl.)}$$

$$\frac{-X, \ X}{\bot} \qquad \text{(spor)}$$

Prototyp

Príklad 1

- zadanie:

$$\{a, \quad a \multimap b, \quad a \Rightarrow c\}$$

- $a \multimap b$ určite skonzumuje a, takže a nemôže byť ztrvácnené a tým pádom sa $a \Rightarrow c$ nevykoná

Príklad 2

- zadanie:

$$\{a, \quad a \multimap b, \quad a \multimap c, \quad (a \multimap c) \Rightarrow ((a \multimap b) \multimap x)\}$$

- pravidlo $a \multimap b$ sa nemôže vykonať kvôli $((a \multimap c) \Rightarrow ((a \multimap b) \multimap x))$

Prototyp

Príklad 1

- zadanie:

$$\{a, \quad a \multimap b, \quad a \Rightarrow c\}$$

- $a \multimap b$ určite skonzumuje a, takže a nemôže byť ztrvácnené a tým pádom sa $a \Rightarrow c$ nevykoná

Príklad 2

- zadanie:

$$\{a, \quad a \multimap b, \quad a \multimap c, \quad (a \multimap c) \Rightarrow ((a \multimap b) \multimap x)\}$$

- pravidlo $a \multimap b$ sa nemôže vykonať kvôli $((a \multimap c) \Rightarrow ((a \multimap b) \multimap x))$

Prototyp

Príklad 3

- zadanie:

$$\{a, \quad a \multimap b, \quad a \multimap c\}$$

- buď sa vyprodukuje b alebo c; bez ďalšej informácie nie je možné preferovať jedno riešenie pred druhým

Príklad 4

- zadanie:

$$\left\{ \begin{array}{c} ((X > Y) \wedge (Y > Z)) \Rightarrow (X > Z) \\ (X > Y) \Rightarrow (X \Rightarrow -Y) \\ a,\ b,\ c,\ d,\ a > b,\ b > c,\ b > d \end{array} \right\}$$

- jedine a môže byť ztrvácnené

Prototyp

Príklad 3

- zadanie:

$$\{a, \quad a \multimap b, \quad a \multimap c\}$$

- buď sa vyprodukuje b alebo c; bez ďalšej informácie nie je možné preferovať jedno riešenie pred druhým

Príklad 4

- zadanie:

$$\left\{ \begin{array}{c} ((X > Y) \wedge (Y > Z)) \Rightarrow (X > Z) \\ (X > Y) \Rightarrow (X \Rightarrow -Y) \\ a,\ b,\ c,\ d,\ a > b,\ b > c,\ b > d \end{array} \right\}$$

- jedine a môže byť ztrvácnené

Čo ďalej

Pomalé

- veľa rôznych poradí, v akom aplikovať pravidlá, aj keď výsledný počet riešení je malý

$$\{a_1, \ldots, a_n, \quad a_1 \multimap b_1, \ldots, a_n \multimap b_n\}$$

- dve rôzne poradia môžu obsahovať to isté odvodenie nejakého termu t

Čo ďalej

Pomalé

- veľa rôznych poradí, v akom aplikovať pravidlá, aj keď výsledný počet riešení je malý

$$\{a_1, \ldots, a_n, \quad a_1 \multimap b_1, \ldots, a_n \multimap b_n\}$$

- dve rôzne poradia môžu obsahovať to isté odvodenie nejakého termu t

Čo ďalej

Zavedenie času

- momentálne každý term je buď skonzumovaný, ale ztrvácnený a ďalej nemôže byť zmenený; dobré na uvažovanie o aktuálnom stave
- stav sa ale môže meniť viac ako len raz; mali by sme byť schopní ovplyvňovať aj ztrvácnené termy

$$\left\{ \begin{array}{c} \mathit{grenade} \multimap (\mathit{goblin} \multimap \mathit{dead\_goblin}) \\ \mathit{goblin} \Rightarrow \mathit{grenade}, \quad \mathit{goblin} \end{array} \right\}$$

Čo ďalej

Zavedenie času

- momentálne každý term je buď skonzumovaný, ale ztrvácnený a ďalej nemôže byť zmenený; dobré na uvažovanie o aktuálnom stave
- stav sa ale môže meniť viac ako len raz; mali by sme byť schopní ovplyvňovať aj ztrvácnené termy

$$\left\{ \begin{array}{c} \mathit{grenade} \multimap (\mathit{goblin} \multimap \mathit{dead\_goblin}) \\ \mathit{goblin} \Rightarrow \mathit{grenade}, \quad \mathit{goblin} \end{array} \right\}$$

Čo ďalej

Obmedzenie na počet použití

- momentálne, ak nejaké pravidlo (tvaru $X \Rightarrow Y$ alebo $X \multimap Y$) je ztrvácnené, tak to môžeme použiť ľubovoľne veľakrát
- niekedy chceme obmedzenie, minimálne rozlíšiť medzi $1\times$ a ľubovoľne veľakrát

$$\left\{ \begin{array}{c} \mathit{sated}\ X \Rightarrow ((\mathit{power}\ X = Y) \Rightarrow (\mathit{power}\ X = Y + 5)) \\ \mathit{sated}\ \mathit{jano}, \quad \mathit{power}\ \mathit{jano} = 5 \end{array} \right\}$$

Čo ďalej

Obmedzenie na počet použití

- momentálne, ak nejaké pravidlo (tvaru $X \Rightarrow Y$ alebo $X \multimap Y$) je ztrvácnené, tak to môžeme použiť ľubovoľne veľakrát
- niekedy chceme obmedzenie, minimálne rozlíšiť medzi $1\times$ a ľubovoľne veľakrát

$$\left\{ \begin{array}{c} \mathit{sated}\ X \Rightarrow ((\mathit{power}\ X = Y) \Rightarrow (\mathit{power}\ X = Y + 5)) \\ \mathit{sated}\ \mathit{jano}, \quad \mathit{power}\ \mathit{jano} = 5 \end{array} \right\}$$

Čo ďalej

Obmedzenie na počet inštancii

- zatiaľ pravidlo s premennými chápeme ako univerzálne kvantifikované ($\forall$)
- niekedy ale chceme vytvoriť len jednu inštanciu ($\exists$), alebo "čo najviac ale nanajvýš 3", ...

$$\left\{ \begin{array}{c} \mathit{lightning\_bolt} \multimap \exists(X : \mathit{creature})(\mathit{deal\_damage}\ 3\ X) \\ \mathit{goblin} : \mathit{creature}, \quad \mathit{elf} : \mathit{creature}, \quad \mathit{sheep} : \mathit{creature} \end{array} \right\}$$

Čo ďalej

Obmedzenie na počet inštancii

- zatiaľ pravidlo s premennými chápeme ako univerzálne kvantifikované ($\forall$)
- niekedy ale chceme vytvoriť len jednu inštanciu ($\exists$), alebo "čo najviac ale nanajvýš 3", ...

$$\left\{ \begin{array}{l} \mathit{lightning\_bolt} \multimap \exists(X : \mathit{creature})(\mathit{deal\_damage}\ 3\ X) \\ \mathit{goblin} : \mathit{creature}, \quad \mathit{elf} : \mathit{creature}, \quad \mathit{sheep} : \mathit{creature} \end{array} \right\}$$

Čo ďalej

Redukcia termov

- v štandardných logických jazykoch, ak chceme robiť aritmetiku, musíme ju explicitne vyhodnotiť

$$power\ jano = 4 + 5 \qquad \text{(zle)}$$

$$(Z\ is\ 4 + 5) \wedge (power\ jano = Z))) \qquad \text{(dobre)}$$

Zhrnutie

- motivovali sme voľbu jazyka založeného na logike
- zvážili sme existujúce prístupy (spomenuli sme GDL-II)
- návrh a implementácia prototypu, príklady
- čo ďalej
  - efektívnosť
  - expresívnosť

Zhrnutie

- motivovali sme voľbu jazyka založeného na logike
- zvážili sme existujúce prístupy (spomenuli sme GDL-II)
- návrh a implementácia prototypu, príklady
- čo ďalej
  - efektívnosť
  - expresívnosť