APS — Lógica e Matemática Discreta

Giovanna Barros Scalco e Gustavo Nicácio
Professor: Felipe Resina

Objetivo

Aplicar os conceitos de Lógica de Primeira Ordem para representar formalmente o universo do jogo Undertale, utilizando predicados, funções e quantificadores. Em seguida, demonstrar deduções naturais com base nas regras lógicas e, depois, implementar o modelo em Prolog.

Cenário Escolhido: Undertale

No universo de Undertale, humanos e monstros coexistem em um mundo em que as ações dos personagens determinam diferentes rotas narrativas: Pacifista, Neutra e Genocida, em que depende do número de monstros que o personagem decide enfrentar. Para esta atividade pode-se modelar logicamente as relações entre personagens.

Modelagem Lógica

Predicado	Significado
H(x)	x é humano
M(x)	x é monstro
D(x)	x tem determinação
P(x)	x está trilhando a rota pacifista
N(x)	x está trilhando a rota neutra
G(x)	x está trilhando a rota genocida

Relação	Significado

C(x,y)	x é pai/mãe de y
K(x,y)	x matou y
S(x,y)	x poupou y
F(x,y)	x é amigo de y

Constante	**Representa**
f	Flowey
s	Sans
t	Toriel

Fórmulas em Lógica de Primeira Ordem

1. $\forall x(H(x) \land \forall y(M(y) \land K(x, y)) \land \neg \exists z(M(z) \land S(x, z)) \to G(x))$
2. $\forall x(H(x) \land \exists y S(x, y) \to P(x) \lor N(x))$
3. $\forall x(H(x) \land \exists y K(x, y) \to G(x) \lor N(x))$
4. $\forall x(H(x) \land \exists y K(x, y) \land \exists z S(x, z) \to N(x))$
5. $\forall x(D(x) \to H(x) \lor M(x) \lor x = f)$
6. $\forall x(H(x) \land G(x) \land S(x, s) \to K(s, x))$
7. $\forall x((H(x) \land \neg M(x)) \lor (\neg H(x) \land M(x)))$
8. $\forall x \forall y \exists z((C(x, y) \land K(z, y)) \to \neg F(x, z))$
9. $\forall x \forall y(H(x) \land M(y) \land P(x) \to F(y, x))$
10. $\exists x(M(x) \land \forall y(H(y) \land S(x, y)) \to x = t)$

Dedução Natural

Para o caso 7:

$\forall x((H(x) \land \neg M(x)) \lor (\neg H(x) \land M(x)))$ deve chegar em $\forall x H(x) \to \forall x(\neg M(x))$

Se todos são humanos, então nenhum é monstro

1. $\forall x((H(x) \land \neg M(x)) \lor (\neg H(x) \land M(x)))$
2. $\forall x\ H(x)$ sup
3. (x0) H(x0) $\forall$e2
4. $(H(x0) \land \neg M(x0) \lor (\neg H(x0) \land M(x0)$ $\forall$e1
5. $H(x0) \land \neg M(x0)$ sup
6. $\neg M(x0)$ $\land$ e5
7. $\neg H(x0) \land M(x0)$ sup
8. $\neg H(x0)$ $\land$ e7
9. $M(x0)$ $\land$ e7

0. H(x0)

1. $\neg H(x0)$ $\neg$ 8,10

2. $\neg M(x0)$ $\neg$e

3. $\neg M(x0)$ $\lor$e3 (5-6)(7-12)

4. $\forall x \neg M(x)$ $\forall$i (3-16)

15. $\forall x\ H(x) \to \forall x \neg M(x)$ -> i (2-17)

16. $\forall x H(x) \to \forall x \neg M(x)$

Para o caso 8:

$\forall x \forall y \exists z((C(x,y) \land K(z,y)) \to \neg F(x,z))$

1. $\forall x \forall y((C(x,y) \land K(f,y)) \to \neg F(x,f))$
2. $\exists x(C(t,x) \land K(f,x))$
3. $C(t,c) \land K(f,c)$ ($\exists$e, c)
4. $(C(t,c) \land K(f,c)) \to \neg F(t,f)$ ($\forall$e 1)
5. $C(t,c)$ ($\land$e 3)
6. $K(f,c)$ ($\land$e 3)
7. $\neg F(t,f)$ ($\to$e 4,3)
8. $\neg F(t,f)$ ($\exists$e 2,3–7)

Implementação em Prolog

github: https://github.com/scalcogigi/aps-matematica-discreta-prolog.git

```
:- discontiguous killed/2.

%%%% CONSTANTES
alias(f, flowey).
alias(s, sans).
alias(t, toriel).

%%%% FATOS

% Humanos
human(frisk).

% Monstros
monster(sans).
monster(papyrus).
monster(asgore).
```

```
monster(toriel).
monster(undyne).
monster(mettaton).
monster(flowey).

% Relações familiares
parent(toriel, frisk).     % Toriel é figura materna de Frisk
parent(asgore, frisk).     % Asgore é figura paterna de Frisk

% Determinação
determined(frisk).
determined(undyne).
determined(flowey).

% Ações
killed(frisk, undyne).
spared(frisk, papyrus).
spared(toriel, frisk).


%%%% HELPERS

exists_killed_monster(X) :- killed(X, Y), monster(Y).
exists_spared_monster(X) :- spared(X, Y), monster(Y).

% ∀y(M(y) ∧ K(x,y))
killed_all_monsters(X) :-
  \+ ( monster(Y), \+ killed(X, Y) ).

% ¬∃z(M(z) ∧ S(x,z))
spared_no_monster(X) :-
  \+ ( monster(Z), spared(X, Z) ).

% ∀y(H(y) ∧ S(x,y))
spares_all_humans_conj(X) :-
  \+ ( human(Y), \+ spared(X, Y) ).

%%%% FÓRMULAS (1-10)

%%%% FÓRMULA 1
```

```
% ∀x( H(x) ∧ ∀y( M(y) ∧ K(x,y) ) ∧ ¬∃z( M(z) ∧ S(x,z) ) → G(x) )
genocidal(X) :-
  human(X),
  killed_all_monsters(X),
  spared_no_monster(X).

%%%% FÓRMULA 2
% ∀x( H(x) ∧ ∃y S(x,y) → P(x) ∨ N(x) )
% (satisfeita indiretamente via definições de P e N)

%%%% FÓRMULA 3
% ∀x( H(x) ∧ ∃y K(x,y) → G(x) ∨ N(x) )
% (satisfeita indiretamente via definições de G e N)

%%%% FÓRMULA 4
% ∀x( H(x) ∧ ∃y K(x,y) ∧ ∃z S(x,z) → N(x) )
neutral(X) :-
  human(X),
  exists_killed_monster(X),
  exists_spared_monster(X).

%%%% REGRAS ÚTEIS
% Pacifista: não matou ninguém e poupou alguém
pacifist(X) :-
  human(X),
  \+ exists_killed_monster(X),
  exists_spared_monster(X).

%%%% FÓRMULA 5
% ∀x( D(x) → H(x) ∨ M(x) ∨ x = f )
determined_is_h_or_m_or_f(X) :-
  determined(X),
  ( human(X)
  ; monster(X)
  ; alias(f, X)
  ).

%%%% FÓRMULA 6
% ∀x( H(x) ∧ G(x) ∧ S(x, s) → K(s, x) )
killed(sans, X) :-
```

```
  human(X),
  genocidal(X),
  spared(X, S).

%%%% FÓRMULA 7
% ∀x( (H(x) ∧ ¬M(x)) ∨ (¬H(x) ∧ M(x)) )
integrity_human_not_monster(X) :- human(X), \+ monster(X).
integrity_monster_not_human(X) :- monster(X), \+ human(X).

%%%% FÓRMULA 8
% ∀x∀y∃z( C(x,y) ∧ K(z,y) → ¬F(x,z) )
not_friend(X, Z) :- parent(X, Y), killed(Z, Y).
not_friend(Z, X) :- parent(X, Y), killed(Z, Y).

%%%% FÓRMULA 9
% ∀x∀y( H(x) ∧ M(y) ∧ P(x) → F(y,x) )
friend(Y, X) :-
  human(X),
  monster(Y),
  pacifist(X),
  \+ not_friend(Y, X).

%%%% FÓRMULA 10
% ∃x( ∀y( M(x) ∧ H(y) ∧ S(x, y) ) → x = t )
is_toriel(X) :-
  monster(X),
  spares_all_humans_conj(X),
  alias(t, X).


%%%% CONSULTAS DE EXEMPLO

% 1. Ver rota de Frisk (fórmulas 1-4)
% ?- route(frisk, R).
route(X, genocidal) :- genocidal(X).
route(X, pacifist)  :- pacifist(X).
route(X, neutral)   :- neutral(X).

% 2. Integridade de tipos (fórmula 7)
```

```
% ?- integrity_human_not_monster(frisk),
integrity_monster_not_human(sans).

% 3. Relação de inimizade (fórmula 8)
% ?- not_friend(toriel, frisk).

% 4. Amizade (fórmula 9)
% ?- friend(papyrus, frisk).

% 5. Monstro que poupa todos os humanos (fórmula 10)
% ?- is_toriel(toriel).
```

O programa foi estruturado em quatro partes principais:

Constantes e fatos: definem humanos, monstros, relações familiares e ações do jogo;

Predicados auxiliares: funções para checar condições, como monstros mortos ou poupados;

Fórmulas previamente definidas: traduções das proposições lógicas apresentadas na APS para a sintaxe de Prolog.

Consultas: exemplos de execução para verificar os comportamentos esperados.

A modelagem segue a lógica proposta, permitindo testar rotas definidas como genocida, neutra ou pacifista

Consultas e resultados esperados

```
% ?- route(frisk, R).                  → R = neutral.
% ?- integrity_human_not_monster(frisk). → true.
% ?- not_friend(toriel, frisk).        → false.
% ?- friend(papyrus, frisk).           → false.
% ?- is_toriel(toriel).                → true
```

- Frisk segue a rota neutra (matou e poupou monstros);

- Humanos e monstros mantém integridade de tipo;

- Toriel não é inimiga de Frisk e poupa todos os humanos;

- O comportamento geral está de acordo com as fórmulas lógicas definidas.

Portanto, o modelo *Prolog* desenvolvido cumpre o objetivo de representar logicamente as relações e ações do universo *Undertale*, demonstrando como a lógica de predicados pode ser aplicada para simular cenários narrativos e verificar formalmente propriedades definidas por fórmulas matemáticas.