Personalized Mathematical Word Problem Generation

Oleksandr Polozov[*] Eleanor O'Rourke[*] Adam M. Smith[*]
Luke Zettlemoyer[*] Sumit Gulwani[‡] Zoran Popović[*]

[*] University of Washington

[‡] Microsoft Research

Word Problems

Suzy is ten years older than Billy, and next year she will be twice as old as Billy. How old is Suzy now?

Evelyn went to the store 8 times last month. She buys 11 stickers each time she goes to the store. How many stickers did Evelyn buy last month?

You attended high school for 4 years. Each year you bought 7 new textbooks. How many textbooks do you have at home now?

Best known way to teach mathematical modelling skills.

Word Problems

- Notoriously difficult as compared to algebra!

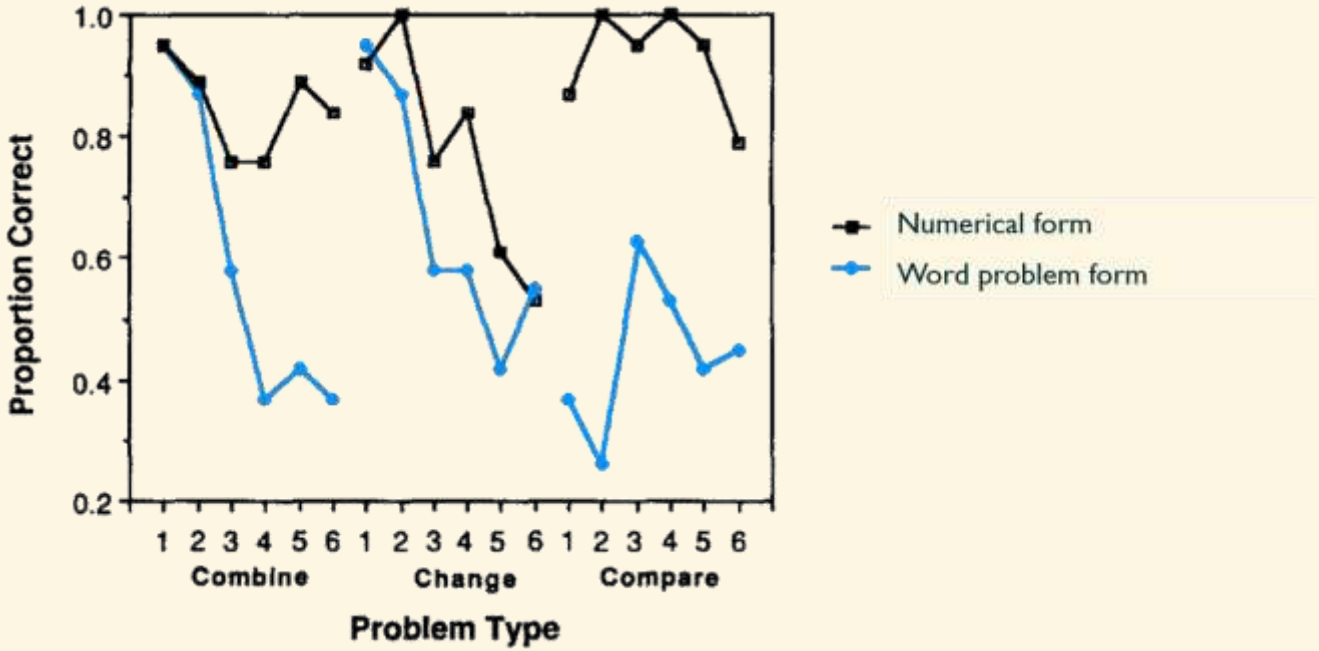
Suzy is ten years older than Billy, and next year she will be twice as old as Billy. How old is Suzy now?

Evelyn went to the store 8 times last month. She buys 11 stickers each time she goes to the store. How many stickers did Evelyn buy last month?

You attended high school for 4 years. Each year you bought 7 new textbooks. How many textbooks do you have at home now?

Word Problems

- Notoriously difficult as compared to algebra!



Suzy is ten years older than Billy, and next year she will be twice as old as Billy. How old is Suzy now?

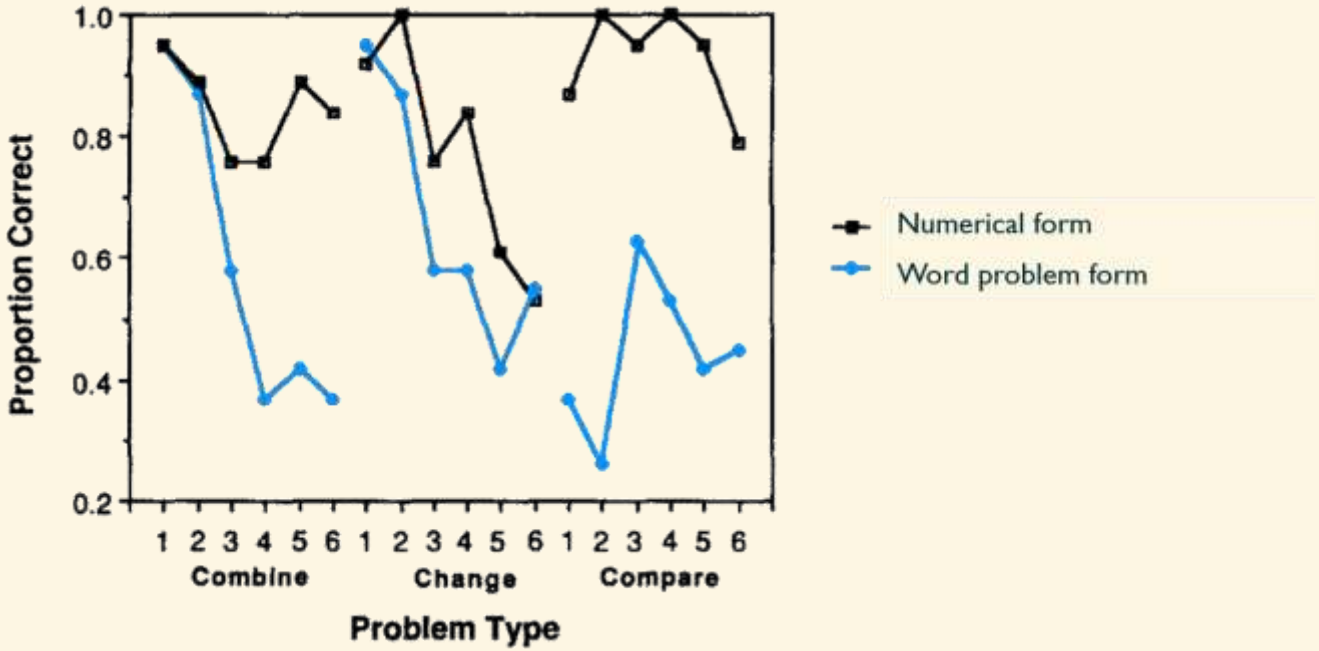
Evelyn went to the store 8 times last month. She buys 11 stickers each time she goes to the store. How many stickers did Evelyn buy last month?

You attended high school for 4 years. Each year you bought 7 new textbooks. How many textbooks do you have at home now?

Cummins, Denise Dellarosa, et al. "The role of understanding in solving word problems." Cognitive psychology 20.4 (1988): 405-438.

Word Problems

- Notoriously difficult as compared to algebra!



Suzy is ten years older than Billy, and next year she will be twice as old as Billy. How old is Suzy now?

Evelyn went to the store 8 times last month. She buys 11 stickers each time she goes to the store. How many stickers did Evelyn buy last month?

You attended high school for 4 years. Each year you bought 7 new textbooks. How many textbooks do you have at home now?

- Perceived as boring, artificial, unconnected to the students' lives $\Longrightarrow$ not learnt

Computer-Aided Pedagogy

- Automatically crafted problem progression:
  - Control over complexity dimensions
  - Per-student personalization
  - Adaptive progression
  - Toolkit for data-driven research
- Enormous design space ⇒ Declarative specification

Workflow

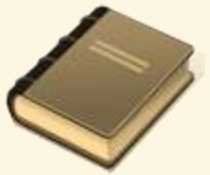
5 problems
Test multiplication:

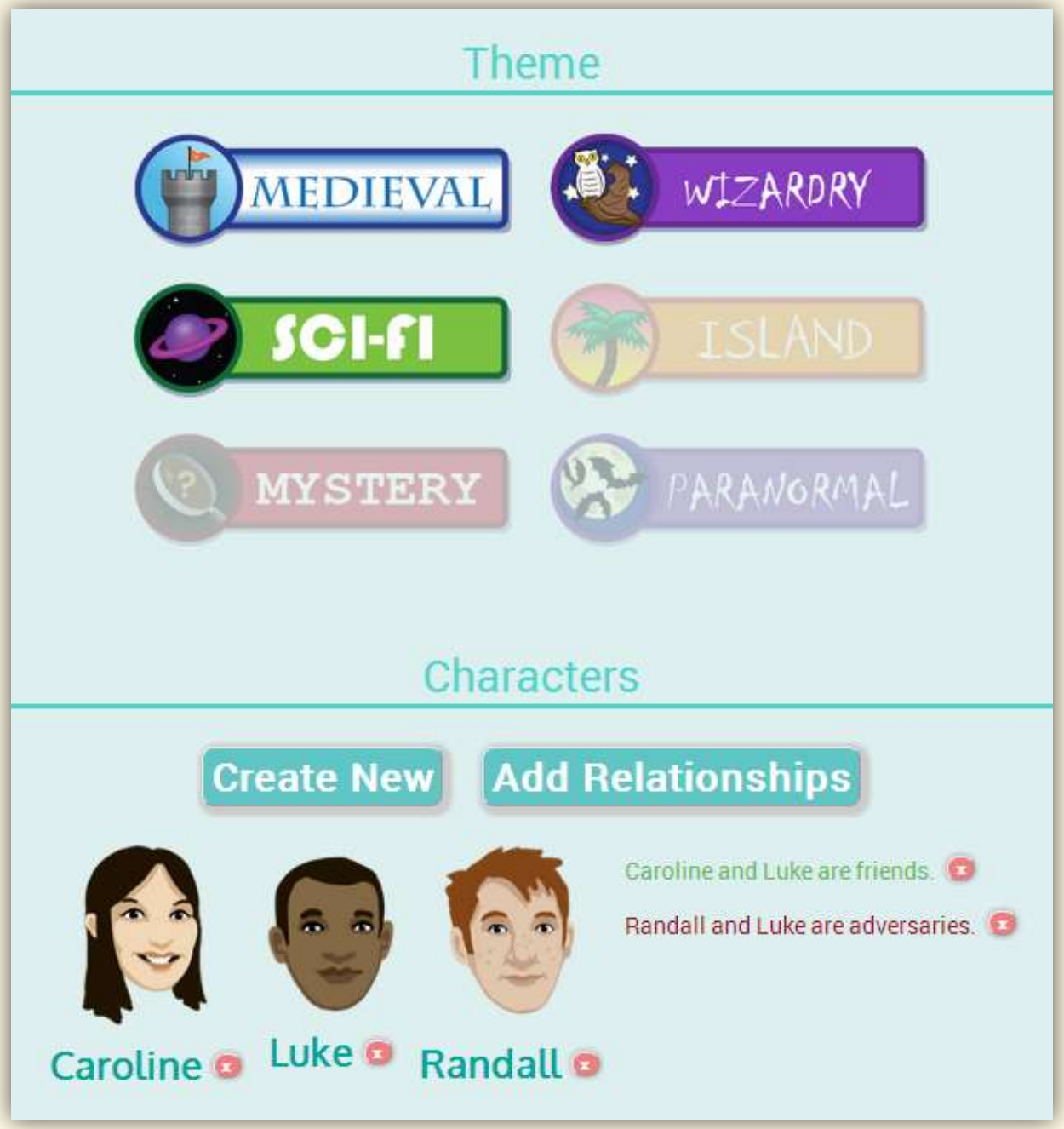
$$x = y \cdot z$$

Time/travel only
Simple language
…

Fantasy/SciFi world
Use me and my friends
as characters

Problem Generator





Duke Randall's countryside consists of 11 towers, surrounded by 3 villages each. He and baron Luke are at war. Luke has already occupied 16 villages with the help of wizard Caroline. How many villages are still unoccupied by the baron?

Workflow

5 problems
Test multiplication:

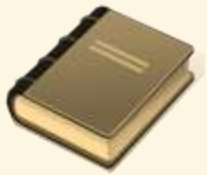
$$x = y \cdot z$$

Time/travel only
Simple language
…

Fantasy/SciFi world
Use me and my friends
as characters

Logic Generation

Language Generation



Problem Generator

Problem Logic Generation

- Plot generation
- Discourse tropes

Natural Language Generation

- Sentence ordering
- Reference resolution

Problem Generator

Problem Logic Generation

- Plot generation
- Discourse tropes

Natural Language Generation

- Sentence ordering
- Reference resolution

Problem Generation =

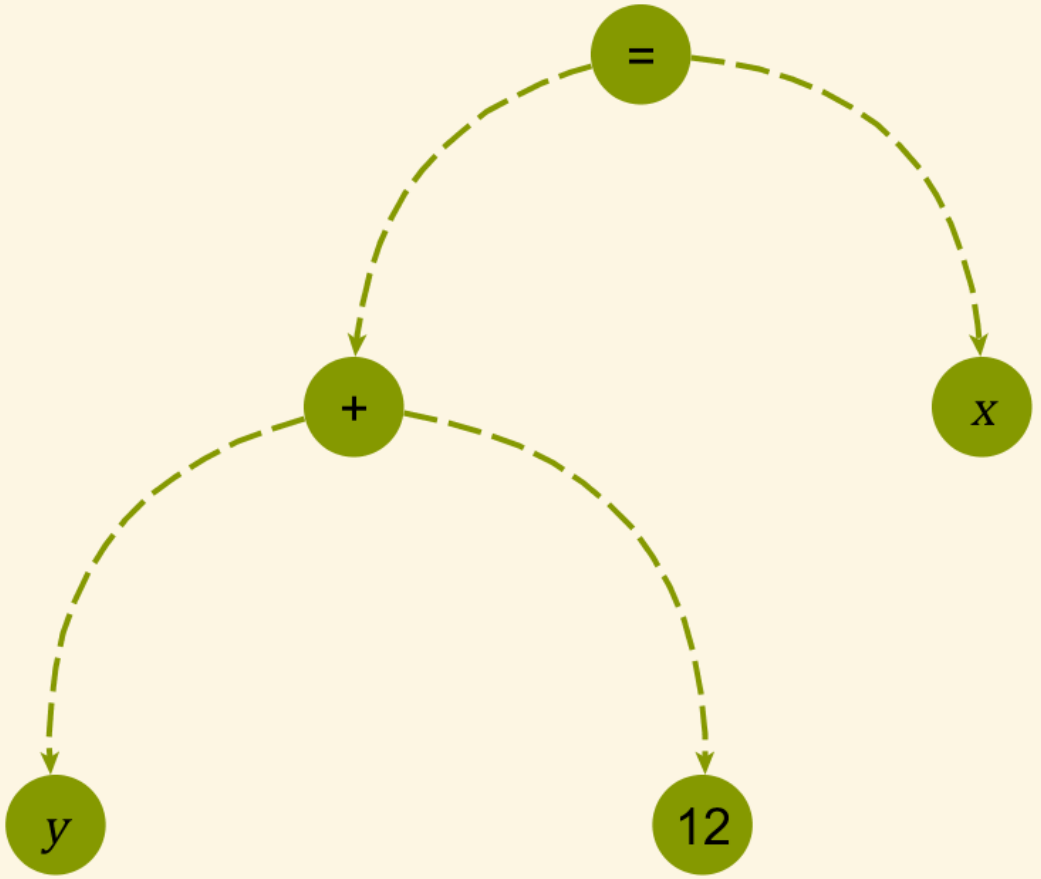
Declaratively constrained synthesis of logical graphs that represent abstract plots

Problem Logic Generation

- Math: addition
- Setting: Fantasy
- Character: Ellie

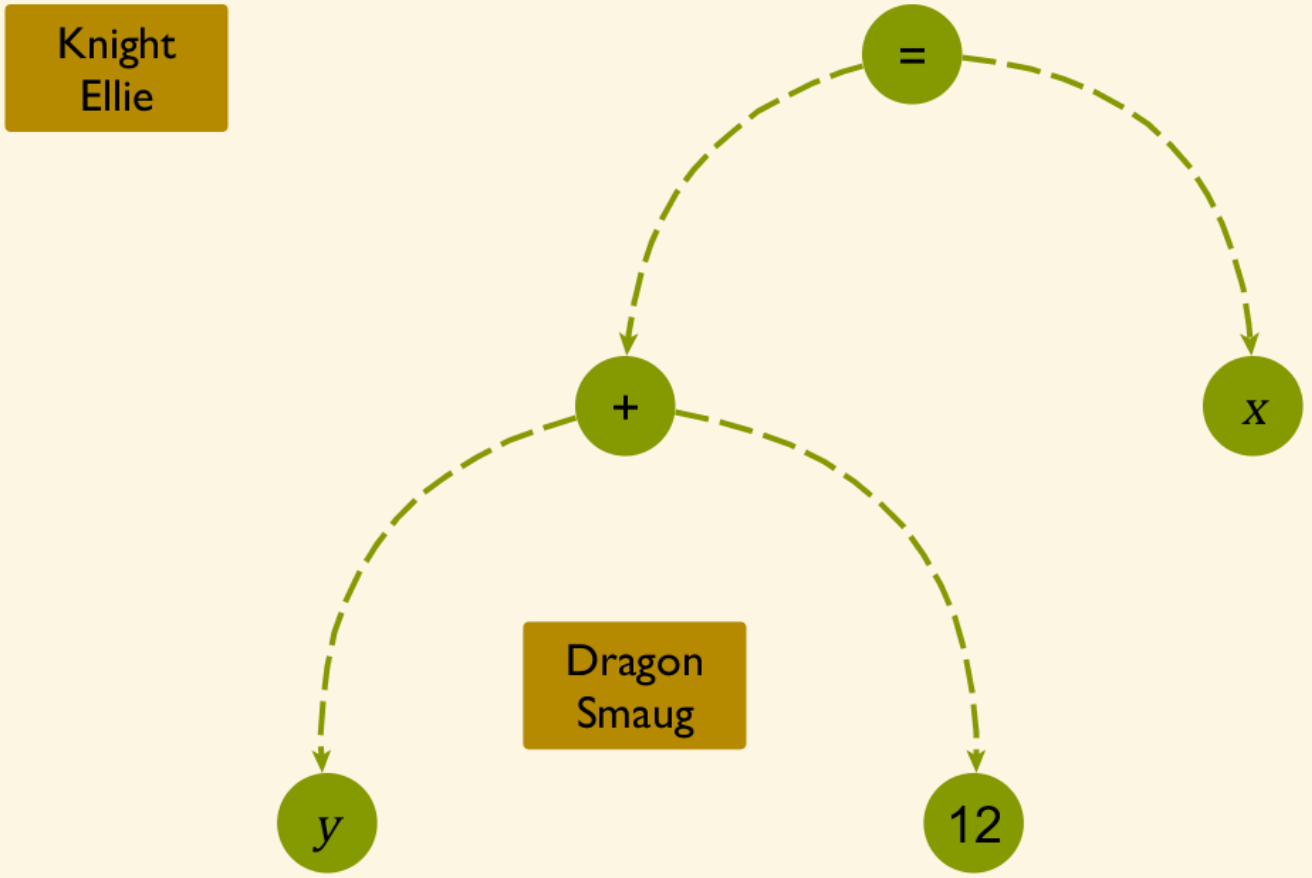
Step 1: Equation

- Math: $x = y + 12$
- Setting: Fantasy
- Character: Ellie



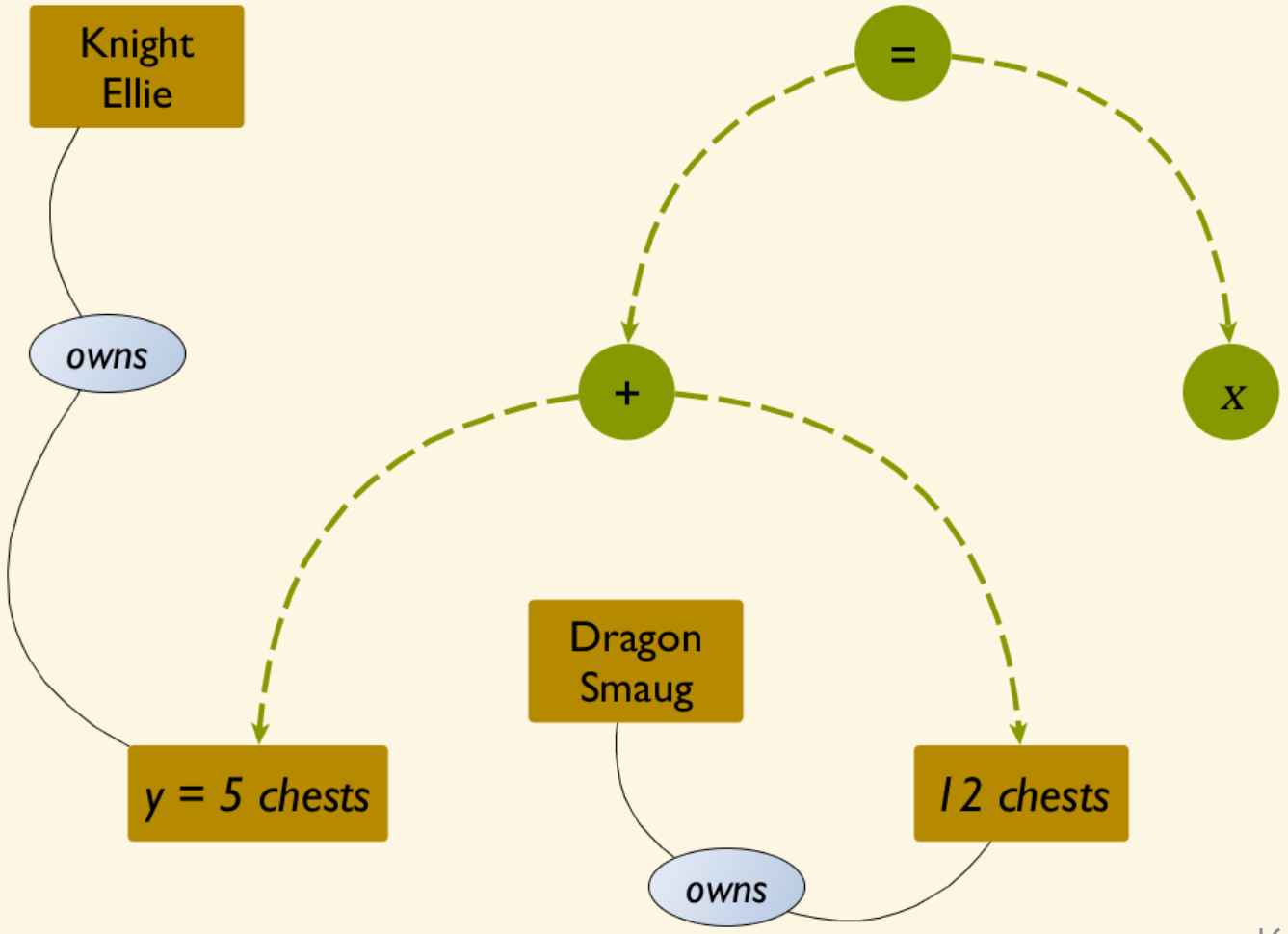
Step 2: Plot Relations

- Math: $x = y + 12$
- Setting: Fantasy
- Character: Ellie



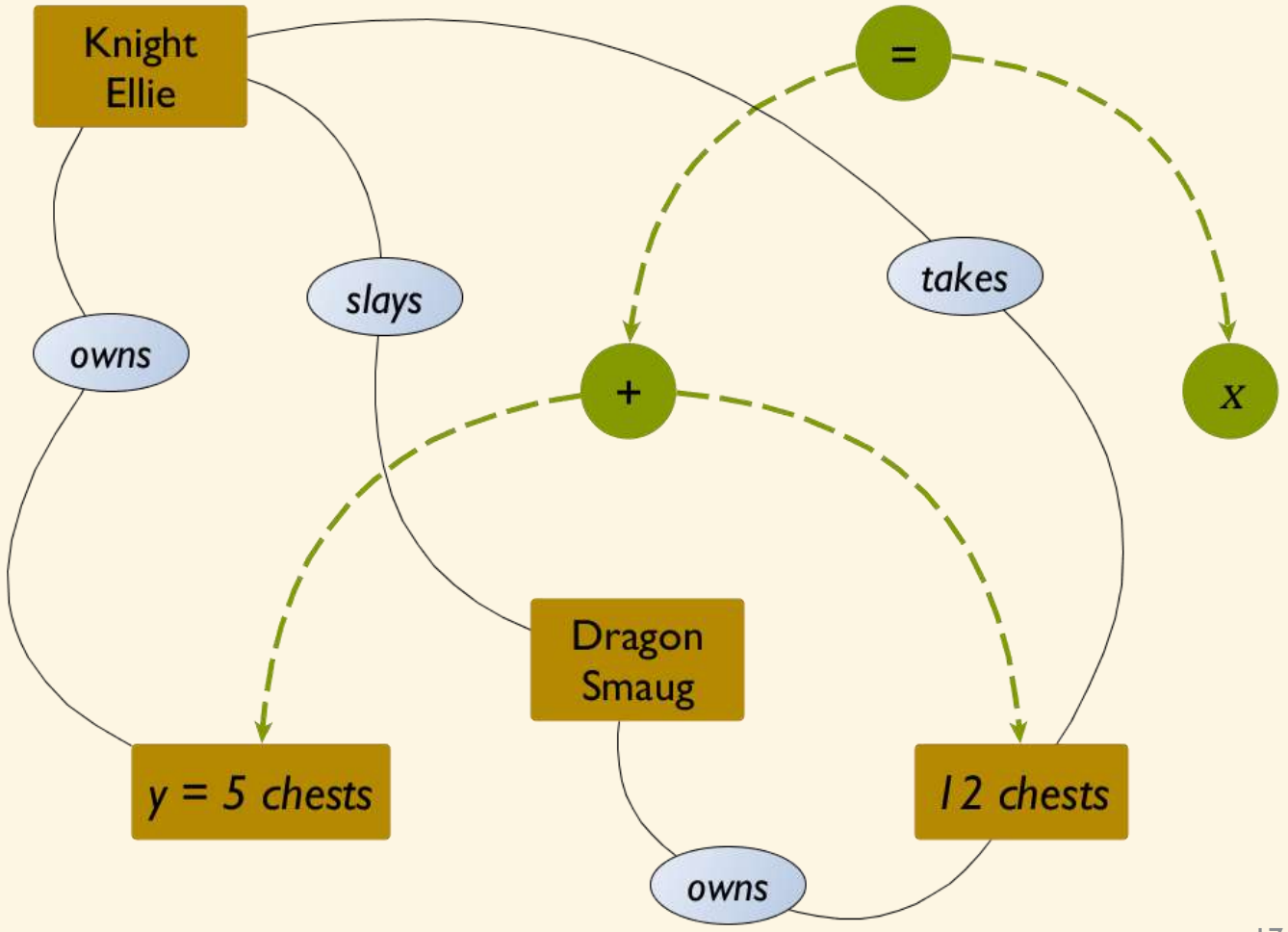
Step 2: Plot Relations

- Math: $x = y + 12$
- Setting: Fantasy
- Character: Ellie



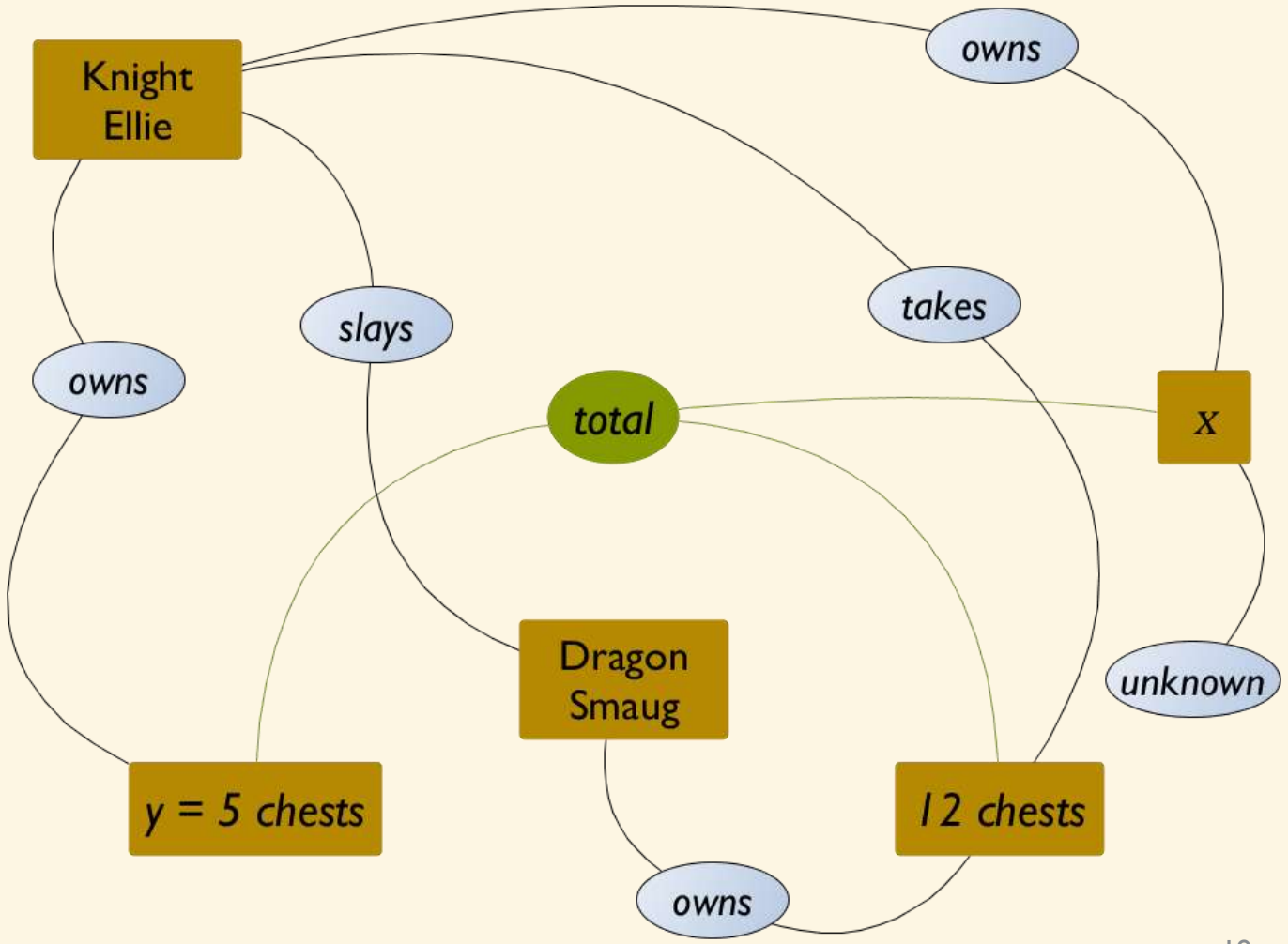
Step 2: Plot Relations

- Math: $x = y + 12$
- Setting: Fantasy
- Character: Ellie



Step 2: Plot Relations

- Math: $x = y + 12$
- Setting: Fantasy
- Character: Ellie



Answer Set Programming

Illustration: Graph Coloring

problem instance

```
node(a). node(b).
node(c). node(d).

edge(a, b). edge(b, c).
edge(a, c). edge(c, d).

color(red).
color(blue).
color(green).
```

problem encoding

```
1 { assign(N, C): color(C) } 1 ← node(N).
```

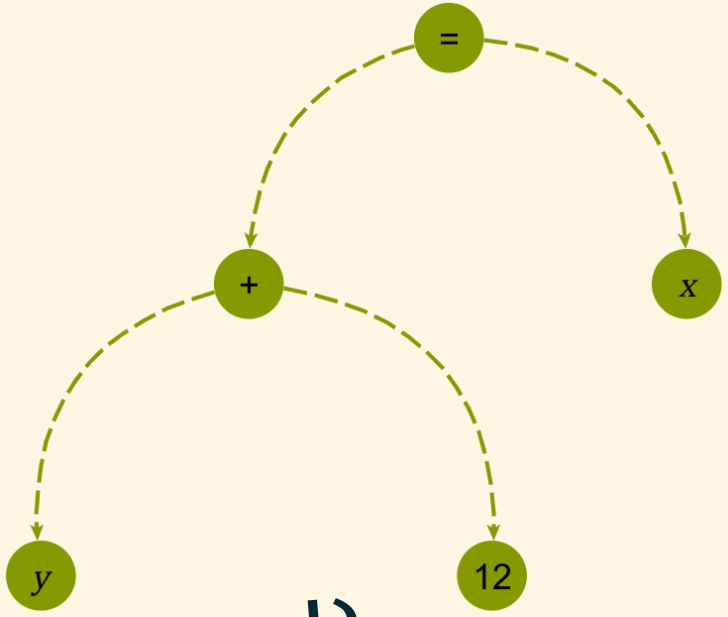
For each node N: nondeterministically pick and assign exactly 1 color C among all existing colors.

If nodes N_1 and N_2 form an edge, they should never be assigned the same color C.

```
← edge(N1, N2),
  assign(N1, C),
  assign(N2, C).
```

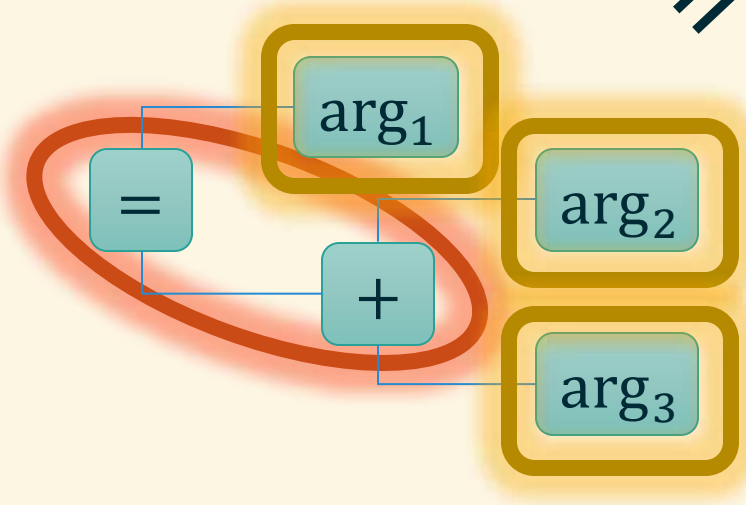
Ontology

```
% Type TWarrior <: TPerson belongs to a fantasy setting.
type(setting(fantasy), t_warrior, t_person).

% Relation Slays(slayer: TWarrior, victim: TMonster) belongs to a fantasy setting.
relation(setting(fantasy), r_slays(t_warrior, t_monster)).

% Arguments slayer and victim in Slays relation can only be adversaries in the plot.
only_relationship(r_slays, adversary(1, 2)).

% TotalCount(total: TCountable, count1: TCountable, count2: TCountable)
relation(setting(common), r_total_count(t_countable, t_countable, t_countable)).


% TotalCount mathematically represents the tree “total = count1 + count2”.
math_skeleton(r_total_count, eq(1, plus(2, 3))).
```

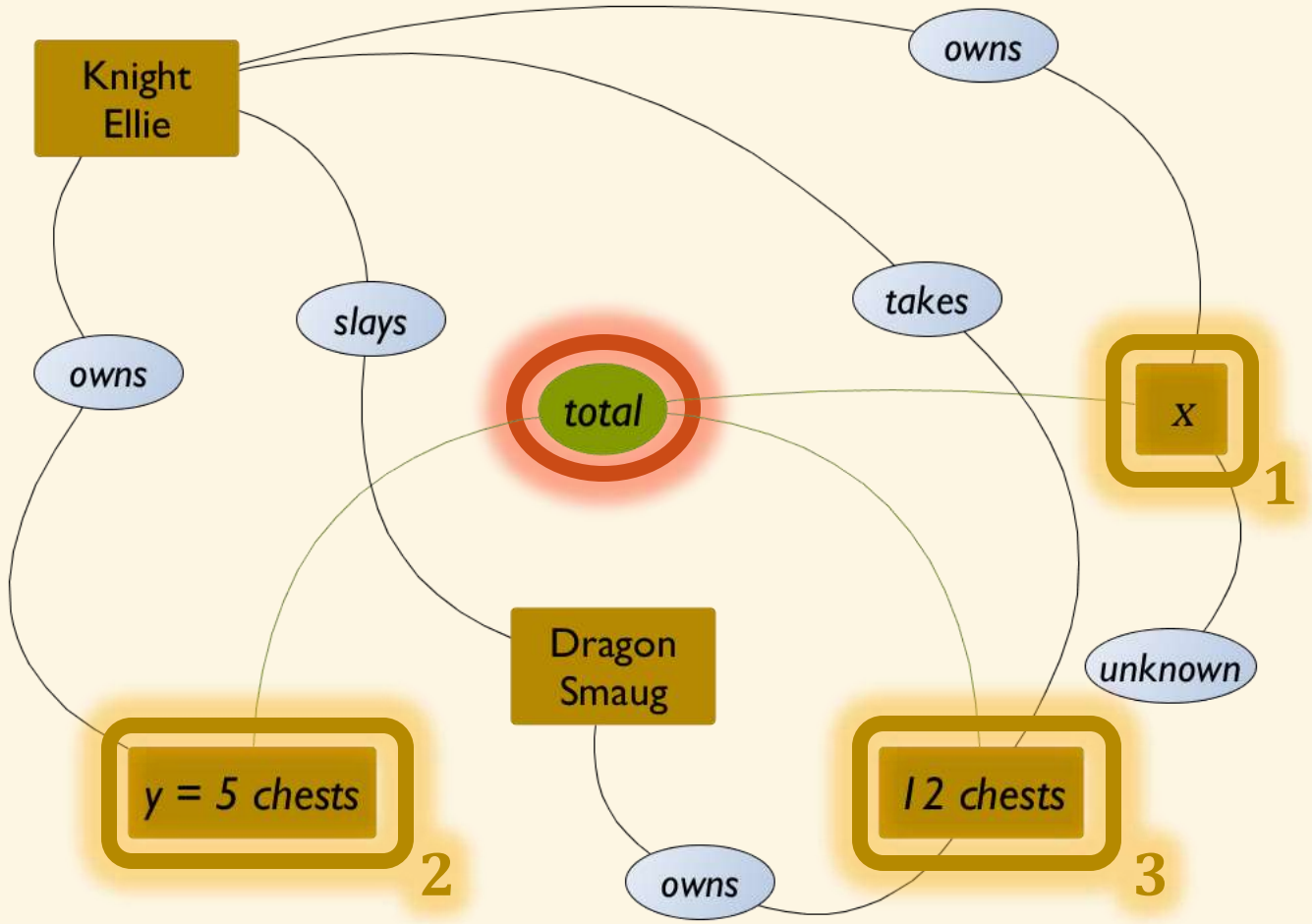


$\wr$

Total:



π



Relation $\simeq$ Equation
Fact $\vDash$ Relation
$\Longrightarrow$ Fact $\vDash$ Equation

Ontology helps us generate *plausible situations*

…but *plausible situation ≠ engaging narrative!*

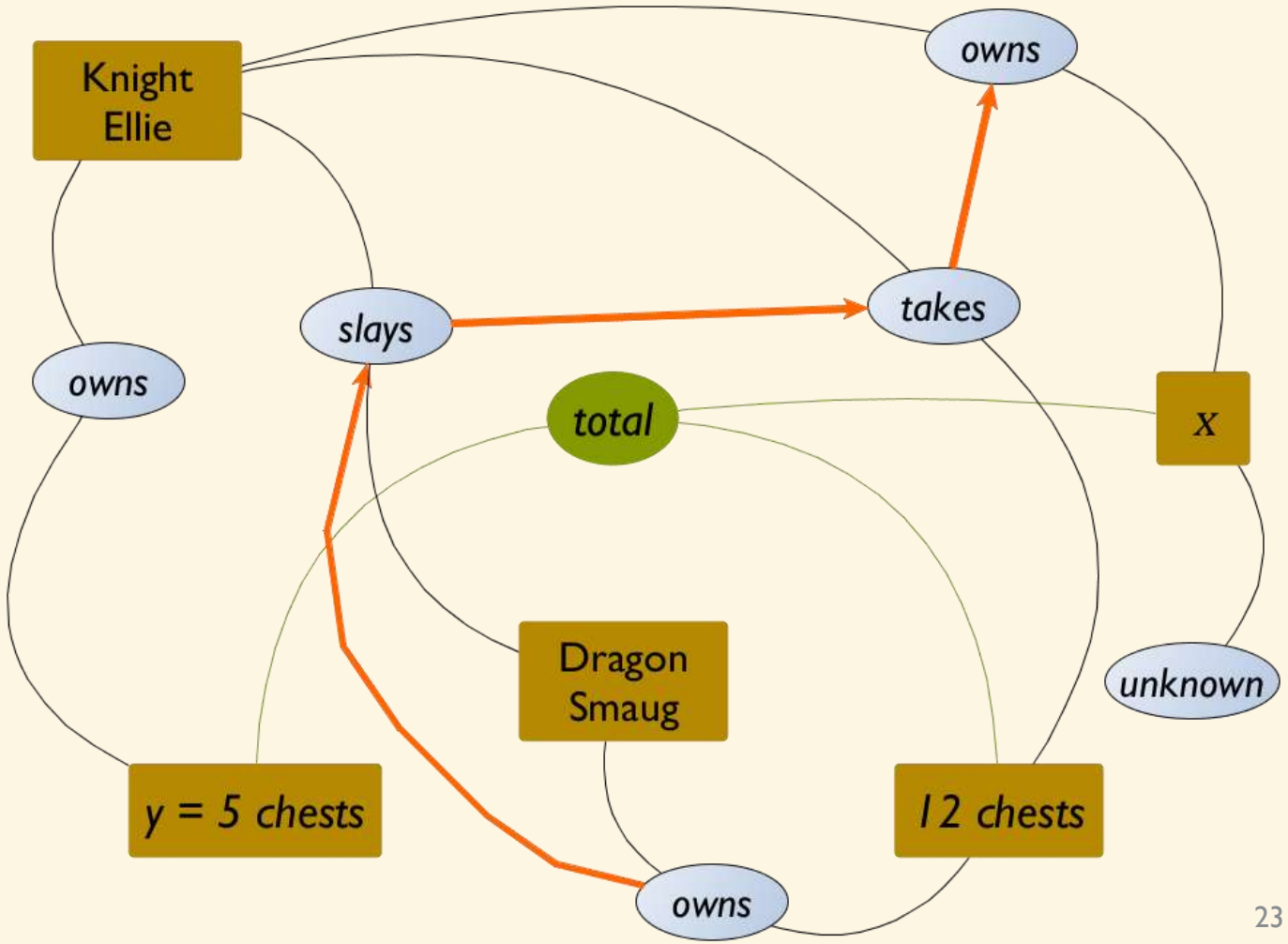
# of satisfying answer sets: up to 10^9. Most are insensible.

Step 3: Discourse Tropes

- Math: $x = y + 12$
- Setting: Fantasy
- Character: Ellie

Tropes = library constraints:

- "Whenever A slays B, A gets everything B had."
- "Whenever A acquires C, A adds C to her possessions."
- "If A is slain, it happens after all her other actions."



Step 3: Discourse Tropes

"A warrior slays a monster only if the monster has some treasures."

$$\forall m, w\colon \text{Slays}(m, w) \implies [\exists t\colon \text{Owns}(m, t)]$$

```
discourse(
  forall( vars(m, w),
          premise(r_slays(w, m)),
          exists( vars(t),
                  conclusion(r_owns(m, t))))).
```

Discourse trope validation

$$\exists \text{ graph } \mathcal{G} = \langle \mathcal{E}, \mathcal{F} \rangle: \quad \text{Valid}(\mathcal{G}) \ \wedge \ \text{Fits}(\mathcal{G}, Reqs) \wedge$$

$$\forall \text{ entities } \vec{x} \subset \mathcal{E}: \ \Phi(\vec{x}) \implies [\exists \vec{y} \subset \mathcal{E}: \ \Psi(\vec{x}, \vec{y})]$$

Discourse trope validation

$$\exists \text{ graph } \mathcal{G} = \langle \mathcal{E}, \mathcal{F} \rangle: \quad \text{Valid}(\mathcal{G}) \ \wedge \ \text{Fits}(\mathcal{G}, Reqs) \wedge$$

$$\left.\begin{array}{l} \forall \text{ entities } \vec{x} \subset \mathcal{E}: \ \Phi_1(\vec{x}) \implies [\exists \vec{y} \subset \mathcal{E}: \ \Psi_1(\vec{x}, \vec{y})] \wedge \\ \vdots \\ \forall \text{ entities } \vec{x} \subset \mathcal{E}: \ \Phi_n(\vec{x}) \implies [\exists \vec{y} \subset \mathcal{E}: \ \Psi_n(\vec{x}, \vec{y})] \end{array}\right\} \text{Library}$$

3 Boolean quantifiers (3QBF) $\implies$ Beyond the capabilities of ASP (not in NP)!

Saturation technique

- Consider 2QBF problem: $\forall a, b: \mathrm{Acquires}(a, b) \rightarrow \mathrm{Owns}(a, b)$
  - ➢ Eliminated innermost $\exists$ by skolemization (polynomial blowup only)
- Apply *disjunctive ASP:* $p_1 \vee \cdots \vee p_k \leftarrow q$.
- Disjunctive ASP has *subset minimality semantics:*

If both M_1 and M_2 are valid answer sets and $M_1 \subset M_2$
then never return M_2

[Eiter, Ianni, Krennwallner 2009]

Saturation technique

var(a). var(b).

```
discourse( forall( vars(a, b),
  premise( implies(acquires(a, b),
                   owns(a, b))))).

bind(V, E): entity(E) ← var(V).

sat(Xs, Tr) ← …

valid ← discourse(Xs, Tr), sat(Xs, Tr).

bind(V, E) ← valid, var(V), entity(E).

← not valid.
```

[Eiter, Ianni, Krennwallner 2009]

Saturation technique

```
discourse( forall( vars(a, b),
  premise( implies(acquires(a, b),
                   owns(a, b))))).

bind(V, E): entity(E) ← var(V).

sat(Xs, Tr) ← …

valid ← discourse(Xs, Tr), sat(Xs, Tr).

bind(V, E) ← valid, var(V), entity(E).

← not valid.
```

(Disjunctively) assign each formal variable ("a" & "b") to some entity in the graph

[Eiter, Ianni, Krennwallner 2009]

Saturation technique

```
discourse( forall( vars(a, b),
  premise( implies(acquires(a, b),
                   owns(a, b))))).

bind(V, E): entity(E) ← var(V).

sat(Xs, Tr) ← …

valid ← discourse(Xs, Tr), sat(Xs, Tr).

bind(V, E) ← valid, var(V), entity(E).

← not valid.
```

Check whether the trope Tr is satisfied under the current variable assignment

[Eiter, Ianni, Krennwallner 2009]

Saturation technique

```
discourse( forall( vars(a, b),
  premise( implies(acquires(a, b),
                   owns(a, b))))).

bind(V, E): entity(E) ← var(V).

sat(Xs, Tr) ← …

valid ← discourse(Xs, Tr), sat(Xs, Tr).

bind(V, E) ← valid, var(V), entity(E).

← not valid.
```

If the trope is not satisfied, the assignment is invalid

[Eiter, Ianni, Krennwallner 2009]

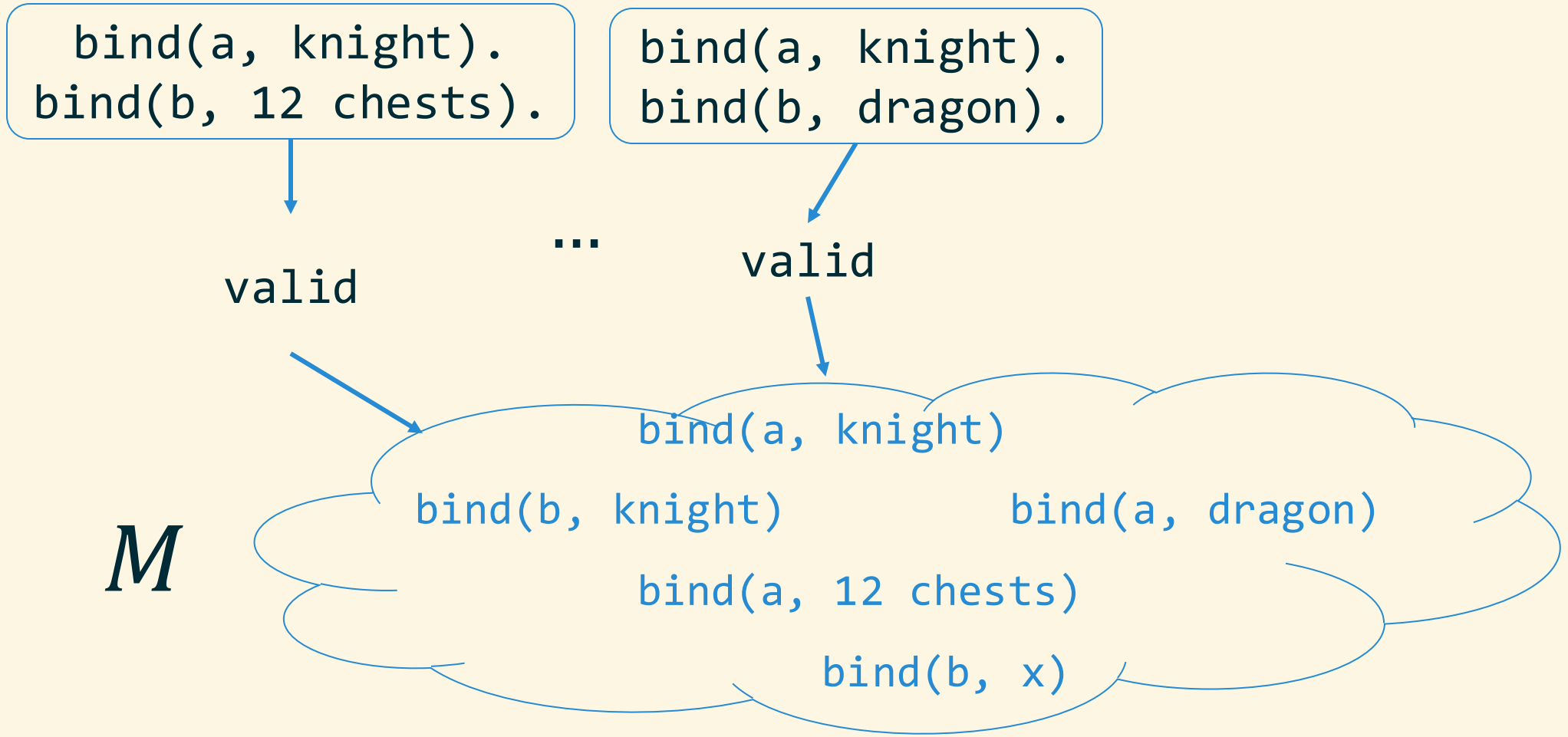
Saturation technique

```
discourse( forall( vars(a, b),
  premise( implies(acquires(a, b),
                   owns(a, b))))).

bind(V, E): entity(E) ← var(V).

sat(Xs, Tr) ← …

valid ← discourse(Xs, Tr), sat(Xs, Tr).

bind(V, E) ← valid, var(V), entity(E).

← not valid.
```

If the trope is satisfied (under 1 assignment only!), *saturate* the answer set: include all possible facts `bind(V, E)` into it.

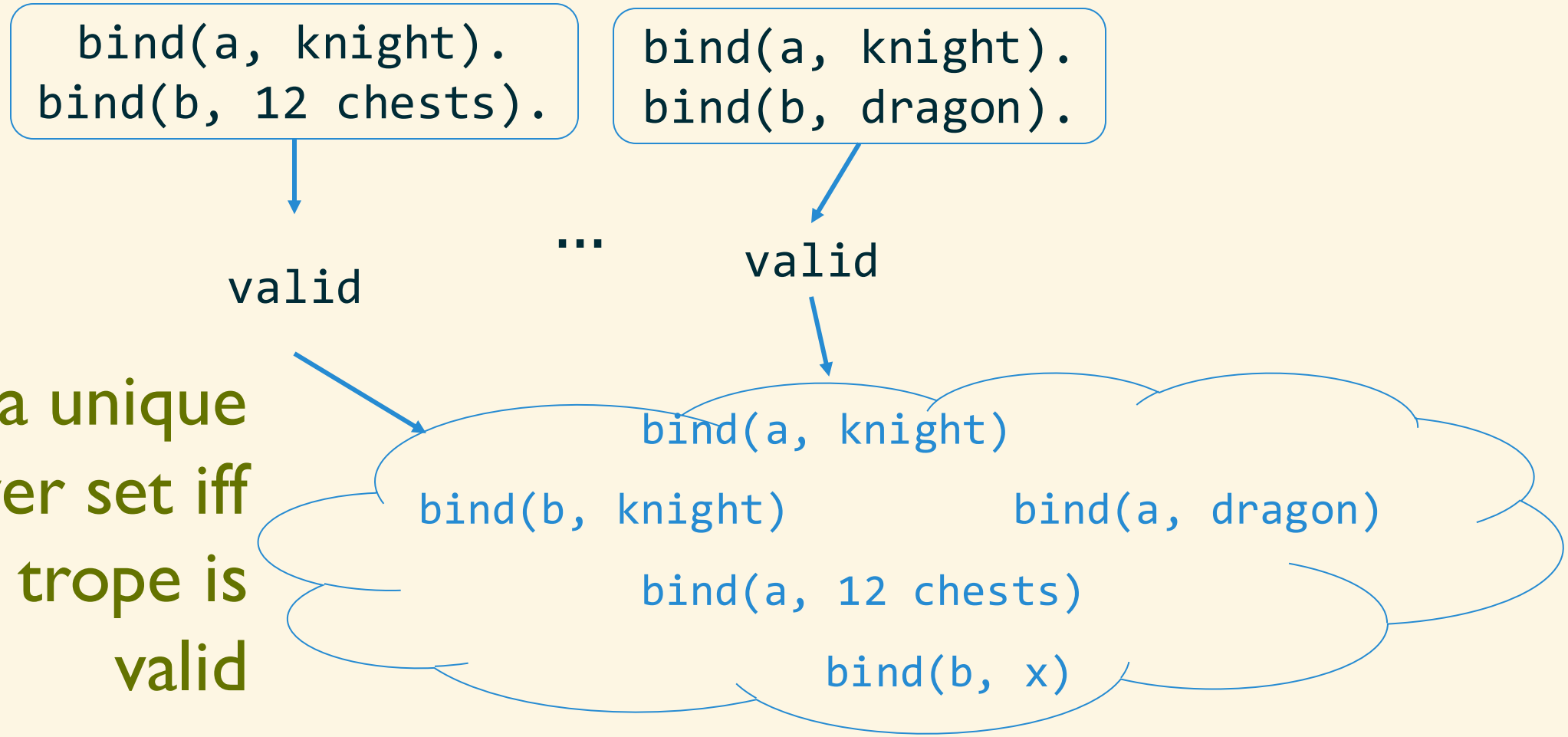
[Eiter, Ianni, Krennwallner 2009]

Saturation technique



[Eiter, Ianni, Krennwallner 2009]

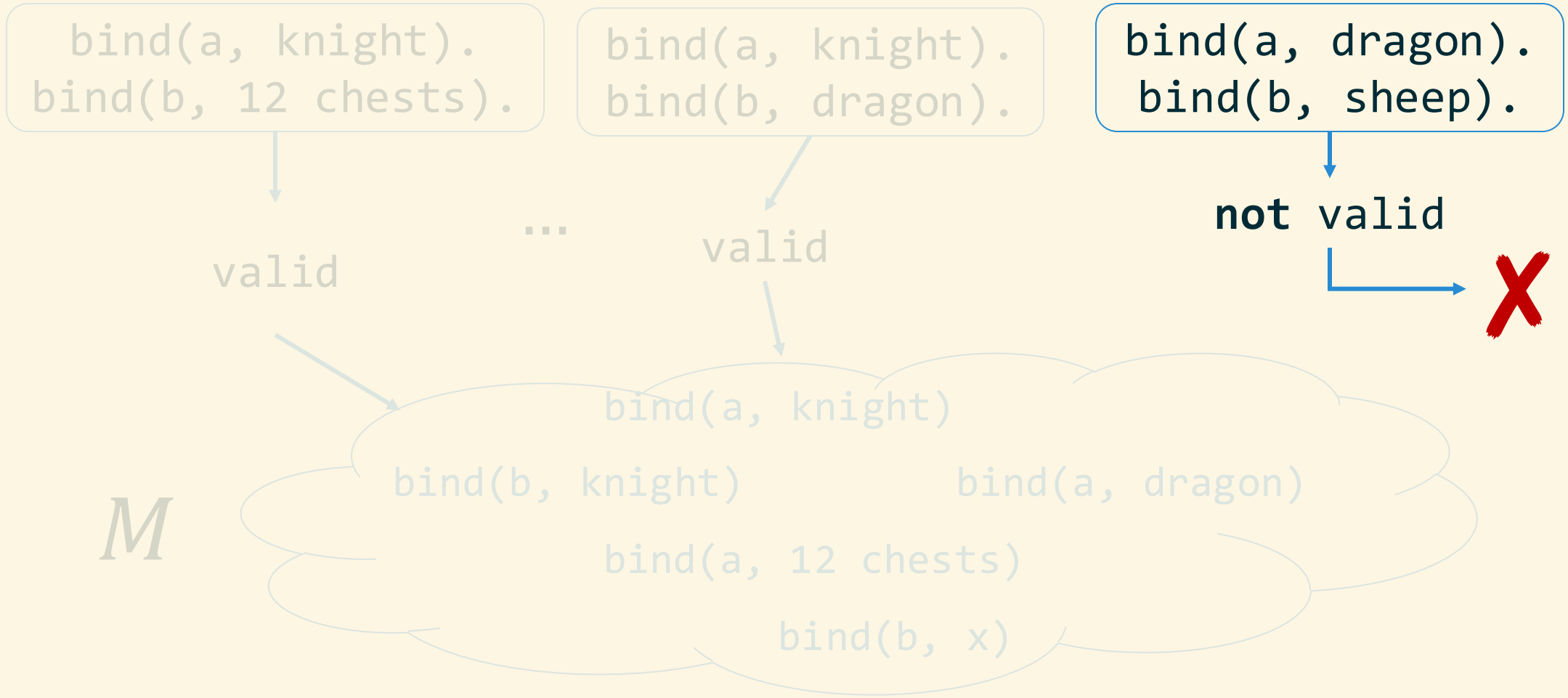
Saturation technique



M is a unique answer set iff the trope is valid

[Eiter, Ianni, Krennwallner 2009]

Saturation technique



[Eiter, Ianni, Krennwallner 2009]

Problem Generator

Problem Logic Generation

- Plot generation
- Discourse tropes

Natural Language Generation

- Sentence ordering
- Reference resolution

Natural Language Generation

Dragon Smaug has 12 chests of treasures.

Knight Ellie has 5 chests of treasures.

Knight Ellie slays Dragon Smaug.

Knight Ellie takes 12 chests of treasures.

How many chests of treasures does Knight Ellie have?

Natural Language Generation: Entity References

Dragon Smaug has 12 chests of treasures.

Knight Ellie has 5 chests of treasures.

She slays the dragon.

Ellie takes his treasures.

How many chests does the knight have?

References should be:

- non-repetitive = "describe the entity with different features every time"
- unambiguous = "differ from entities mentioned previously in at least one feature"

Final problem

Dragon Smaug has 12 chests of treasures.
Knight Ellie has 5 chests of treasures.
She slays the dragon, and takes his treasures.
How many chests does the knight have?

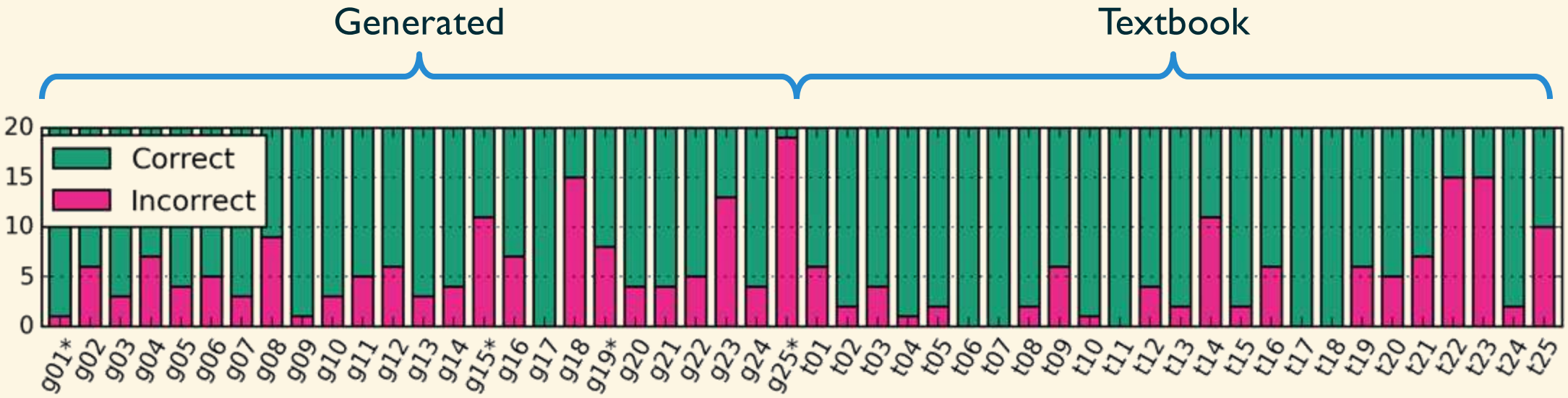
Evaluation

- Focus on *content quality*, not *personalization effects*

- 25 Singapore Math problems vs. 25 autogenerated problems (with equivalent complexity distribution)

- Two MTurk studies, 1000 participants each:
  A. Mathematical applicability (solution time, correctness)
  B. Linguistic aspects (subject-evaluated, Likert scale)

Mathematical applicability



No statistically significant difference in solving times or correctness rates!
(78% for textbook [$\mu = 220\ s$], 73% for generated [$\mu = 232\ s$])

Linguistic comprehensibility

Forced-choice Likert scale ($1 =$ "Strong minus", $4 =$ "Strong plus"):

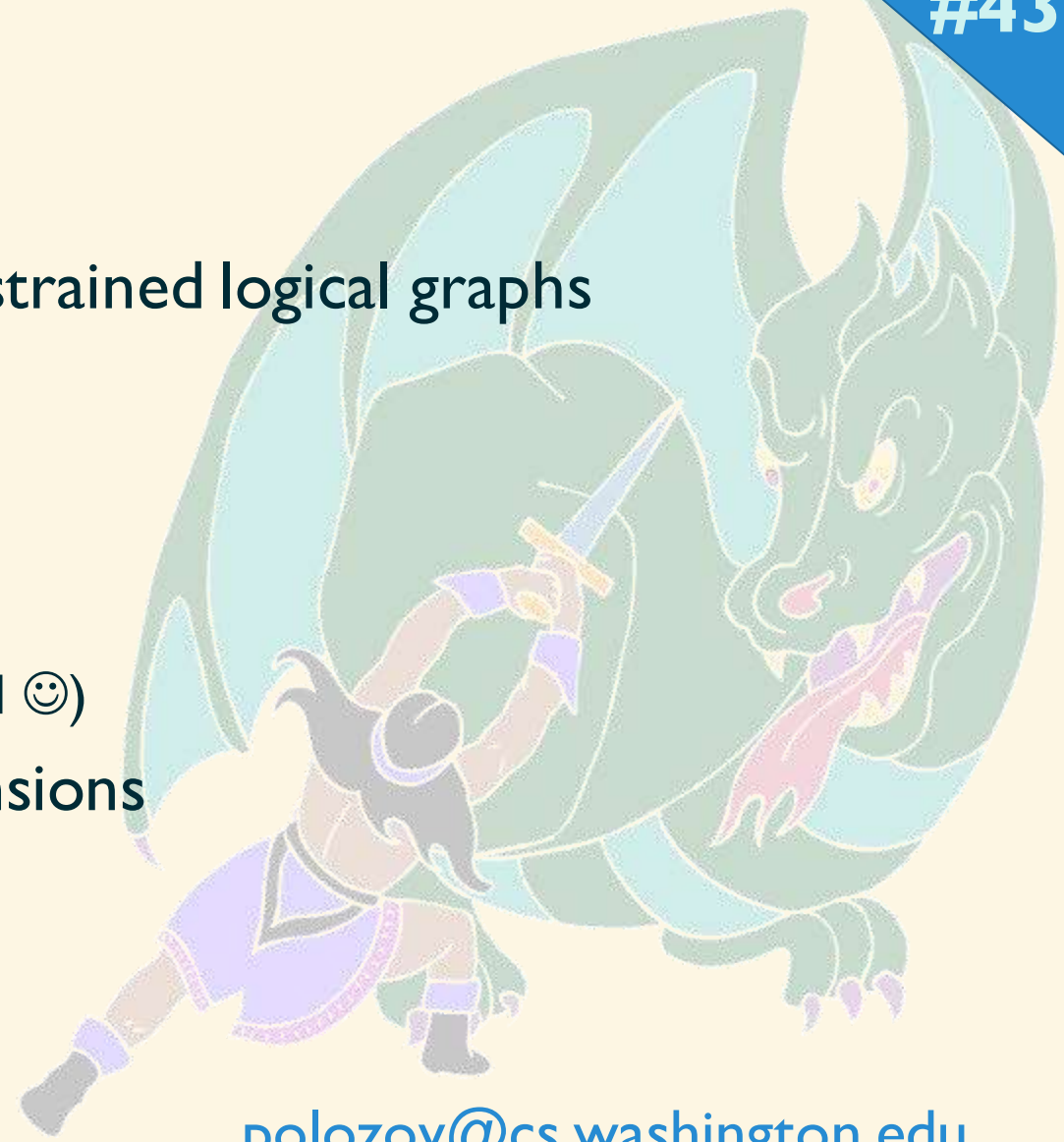
1. How comprehensible is the problem? How well did you understand the plot?
2. How logical/natural is the sentence order?
3. When the problem refers to an actor (e.g. with a pronoun, a name), is it clear who is being mentioned?
4. Do the numbers in the problem fit its story (e.g. it would not make sense for a knight to be 5 years old)?

Expectation: generated problems are noticeably worse (they are generated!). ✓
Goal: they are still comprehensible above a comfortable threshold (mean ≥ 3). ✓

Reality: Mean rating for generated: $\mathbf{3.45 - 3.65}$
Mean rating for textbook: $\mathbf{3.90 - 3.92}$

Summary

- Problem Generation = synthesis of constrained logical graphs
  - Domain-independent
  - Sensible (thanks to discourse tropes)
- State-of-the-art quality problems
  - As solvable as textbook
  - Slightly more artificial language (as expected ☺)
- Total control over the complexity dimensions
  - Customized problem progression
  - Personalization
- What's next? Adaptive curriculum!
- Thank you!



polozov@cs.washington.edu

Backup

Plot generation as Graph isomorphism

```
1 { entity_type(E, T): concrete_type(T) } 1 ← entity(E).
instanceof(E, T) ← entity_type(E, T1), subtype(T1, T).

1 { fact_relation(F, R): relation(R) } 1 ← fact(F).

1 { fact_argument(F, K, E): instanceof(E, T) } 1 ←
      fact_relation(F, R),
      K = 1..@arity(R),
      relation_param_type(R, K, T).

models(Eq, F) ← fact_relation(F, R), math_skeleton(R, S),
                shape_matches(Eq, F, S).

← equation(Eq), #count { F: matches(Eq, F) } == 0.
```

Plot generation as Graph isomorphism

```
1 { entity_type(E, T): concrete_type(T) } 1 ← entity(E).
instanceof(E, T) ← entity_type(E, T1), subtype(T1, T).

1 { fact_relation(F, R)                    } 1 ← fact(F).
1                                            f(E, T) } 1 ←

      relation_param_type(R, K, T).

models(Eq, F) ← fact_relation(F, R), math_skeleton(R, S),
                shape_matches(Eq, F, S).

← equation(Eq), #count { F: matches(Eq, F) } == 0.
```

Entities are *object nodes* in the plot graph.
Pick a single concrete type T for each entity E.

Plot generation as Graph isomorphism

```
1 { entity_type(E, T): concrete_type(T) } 1 ← entity(E).
instanceof(E, T) ← entity_type(E, T1), subtype(T1, T).

1 { fact_relation(F, R): relation(R) } 1 ← fact(F).
1 { fact_argument(F, K, ): instanceof(E, T) } 1 ←
      fact_relation(F, R)
      K = 1..@arity(R),
                                                keleton(R, S),

← equation(Eq), #count { F: matches(Eq, F) } == 0.
```

Facts are *actions nodes* in the plot graph.
For each fact F, pick a single relation R that it represents.

Plot generation as Graph isomorphism

```
1 { entity_type(E, T): concrete_type(T) } 1 ← entity(E).
instanceof(E, T) ← entity_type(E, T1), subtype(T1, T).

1 { fact_relation(F, R): relation(R) } 1 ← fact(F).
1 { fact_argument(F, K, E): instanceof(E, T) } 1 ←
    fact_relation(F, R),
    K = 1..@arity(R),
    relation_param_type(R, K, T).
```

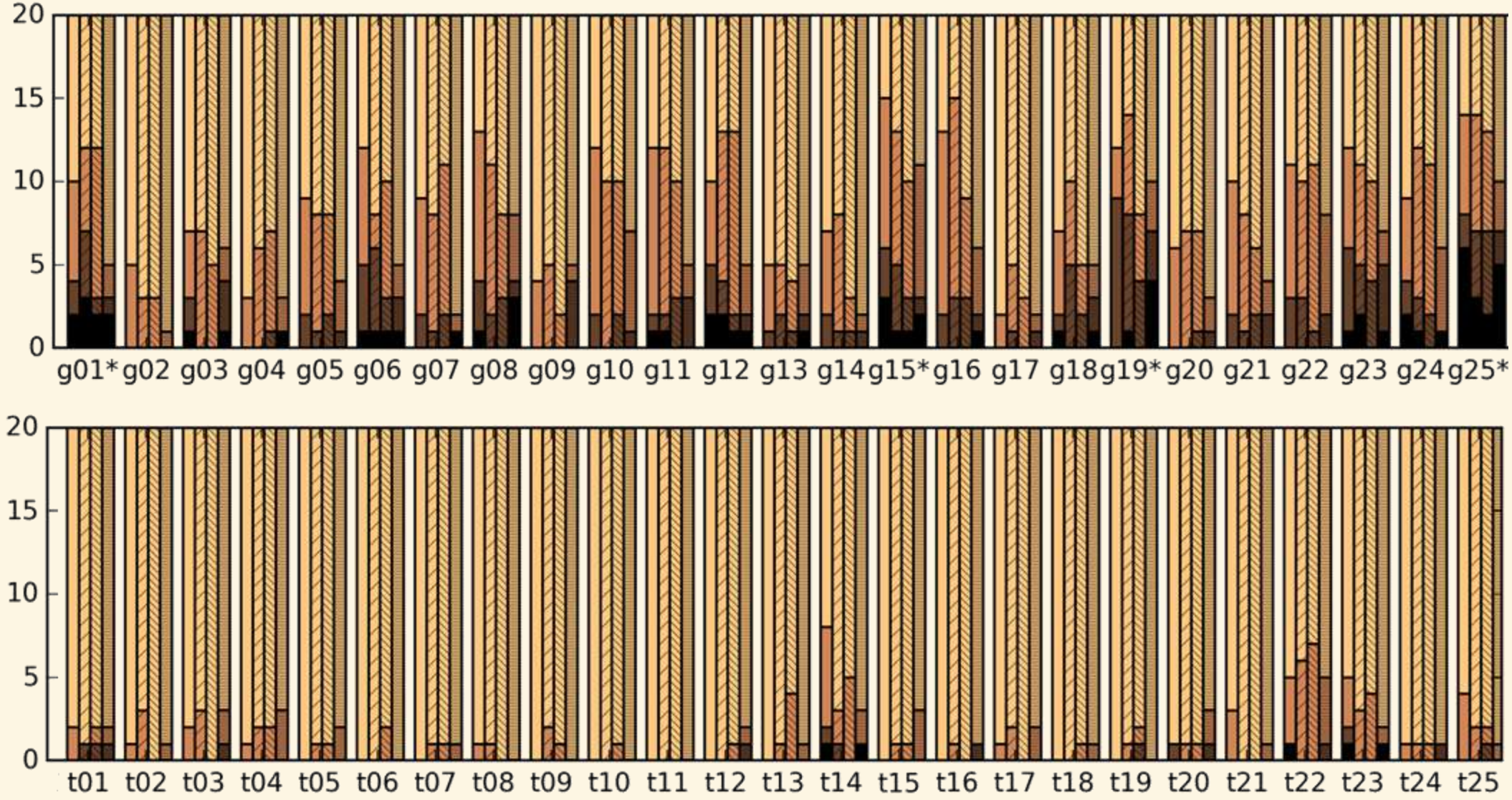
For each fact F representing a k-ary relation R: pick k entities as arguments.
Ensure that they inherit the expected parameter types of R.

Plot generation as Graph isomorphism

A fact F models an equation Eq if it represents a mathematical relation R with a skeleton S that is isomorphic to the equation tree.
Forbid graphs without any facts modelling the equation.

```
1 { fact_argument(F, K, E): instan                    1 ←
      fact_relation(F, R),
      K = 1..@arity(R),
      relation_param_type(R, K, T).

models(Eq, F) ← fact_relation(F, R), math_skeleton(R, S),
                shape_matches(Eq, F, S).

← equation(Eq), #count { F: matches(Eq, F) } == 0.
```

Linguistic comprehensibility



Equation generation

```
node(1..5).     operator(plus; eq).

% Assign an operator and 2 arguments to some nodes.
0 { node_op(N, O): operator(O) } 1 ← node(N).
1 { node_arg(N, K, A): node(A) } 1 ← node_op(N, _), K = 1..2.

root(N) ← node(N), #count { P: node_arg(P, _, N) } == 0.

% Nodes should form a tree with one root, which represents a "=".
← #count { N: root(N) } != 1.
← root(N), not node_op(N, eq).
← node_arg(N, _, A), N > A.
← node(A), #count { N: node_arg(N, _, A) } > 1.

% The equation should match the given math requirements…
```