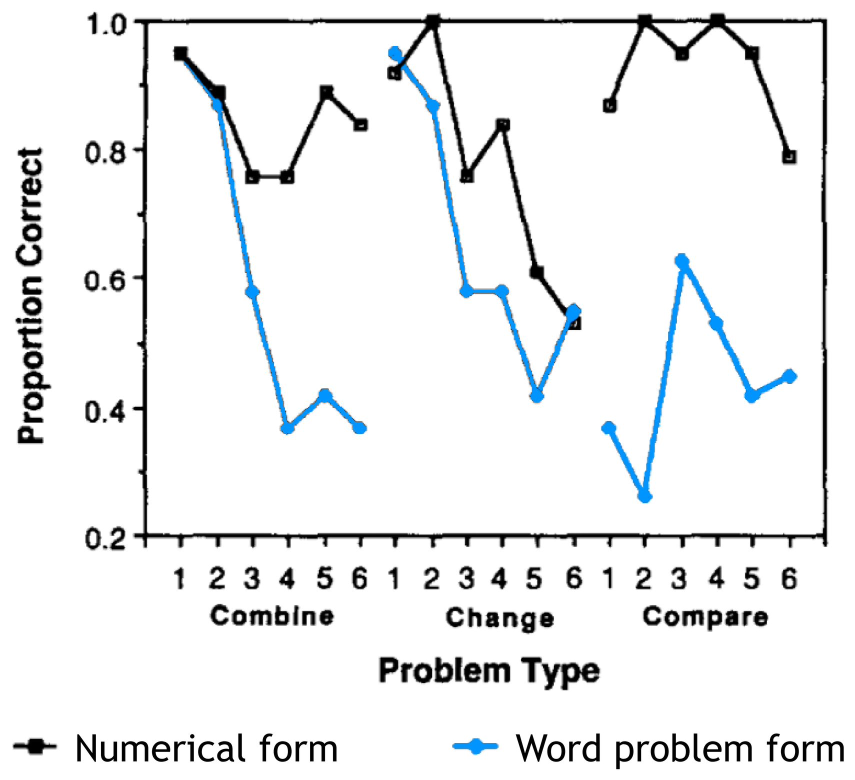
Personalized Mathematical Word Problem Generation

Oleksandr Polozov* Eleanor O'Rourke* Adam M. Smith* Luke Zettlemoyer* Sumit Gulwani† Zoran Popović*
*University of Washington †Microsoft Research Redmond
{polozov, eorourke, amsmith, lsz, zoran}@cs.washington.edu sumitg@microsoft.com

Introduction

Word problems are notoriously difficult for children and adults alike.[1] Many people find them much more difficult than the equivalent symbolic representations (see comparison on the right).[2] This phenomenon is caused by language understanding, conceptual knowledge, discourse comprehension, and other aspects required to build a mental representation of a word problem.[2,3]

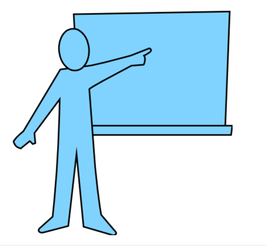
Moreover, many students find word problems unconnected to their lives and artificial.[4] This perception can be altered with problem personalization: individual interest raises understanding and engagement in a problem solving process (which, in turn, increases students' performance).[5] However, personalizing word problems in a textbook is impractical, and would place unreasonable burden on teachers (who would need to be aware of every student's interests).



Our system is a first step to an *ideal pedagogy,* which involves an *individually crafted personalized progression* of word problems:

- **Automatic:** a mathematical model, a plot, and a discourse of a problem are generated automatically from general specifications.
- **Personalized:** students can set preferences for a word problem's setting, characters, and their relationships.
- **Sensible:** we enforce coherence in a synthesized plot using a novel technique called *discourse tropes.*
- **Fit for scaffolding:** varying requirements to different layers of a word problem enables a tutor to scaffold a unique educational progression.

Word problem generation = synthesis of constrained logical graphs + natural language generation



- 1 problem
- Math: addition
- Type: counting
- Simple English

Tutor requirements

Student requirements

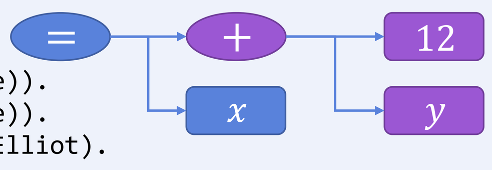
- Fantasy setting
- Characters:
  - girl Ellie
  - boy Smaug
  - adversaries

Logic generation

Our technique uses ***answer set programming (ASP)*** [6] in steps:

1. Equation Generation
   a. *Guess* an equation tree E.
   b. *Deduce* whether math requirements are covered by E.
   c. *Forbid* invalid trees that do not cover the requirements.

```
require_setting(fantasy).
require_math(plus(any, any)).  % "? + ?"
require_character(cAlice, ("Alice", female)).
require_character(cElliot, ("Elliot", male)).
require_relationship(adversary, cAlice, cElliot).
```



2. Plot Generation

Generates a *logical graph* $\mathcal{G}$, which represents a word problem plot that models the equation E:

Definition. A *logical graph* $\mathcal{G}$ is a tuple $\langle \mathcal{E}, \mathcal{F}, \mathcal{C} \rangle$ where:

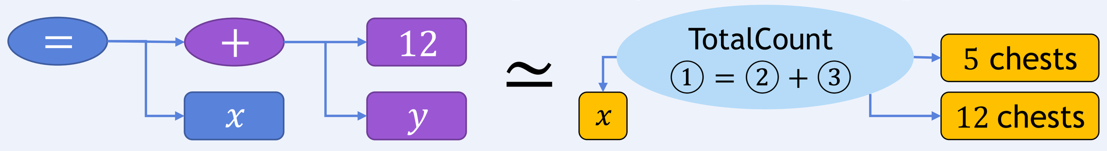
- $\mathcal{E}$ is a set of *entities.* Every entity $e\colon \tau \in \mathcal{E}$ has a corresponding *ontology type* τ. Types form a hierarchy tree, denoted $\tau_1 \preccurlyeq \tau_2$.
- $\mathcal{F}$ is a set of *facts.* Every fact $f \in \mathcal{F}$ has a corresponding *ontology relation* $\mathcal{R} = \text{relation}(f)$. Every relation $\mathcal{R}$ has a set of named *arguments* $\text{args}(\mathcal{R})$. For each fact $f \in \mathcal{F}$, every argument $a\colon \tau_a \in \text{args}(\text{relation}(f))$ is associated with an entity $e\colon \tau_e \in \mathcal{E}$ such that $\tau_e \preccurlyeq \tau_a$, written as $f = \mathcal{R}(e_1, \dots, e_n)$.
- $\mathcal{C}$ is a set of *temporal* (T) or *causal* (C) fact connectives. A connective $c \in \mathcal{C}$ is a tuple $f_1 \Rightarrow_t f_2$ where tag $t \in \{\text{T}, \text{C}\}$.

Example. $\mathcal{E} = \{k\colon$ Knight Ellie, $d\colon$ Dragon Smaug, $c_k\colon$ 5 chests, $c_d\colon$ 12 chests, $c_u\colon$ $x\}$

$\mathcal{F}$: { $\text{Owns}_1(k, c_k)$, $\text{Owns}_2(d, c_d)$, $\text{Slays}(k, d)$, $\text{Acquires}(k, c_d)$, $\text{TotalCount}(c_u, c_k, c_d)$, $\text{Owns}_3(k, c_u)$, $\text{Unknown}(c_u)$ }

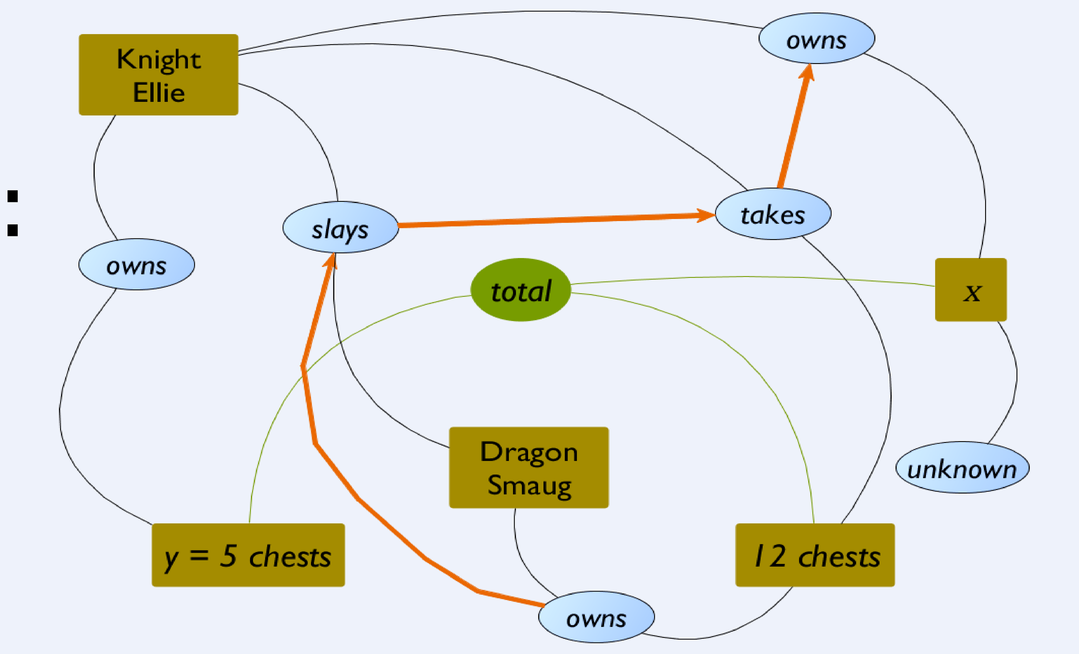
$\mathcal{C}$: $\{\text{Owns}_1 \Rightarrow_\text{T} \text{Slays},\ \text{Owns}_2 \Rightarrow_\text{T} \text{Slays},\ \text{Slays} \Rightarrow_\text{C} \text{Acquires}\}$

Some relations $\mathcal{R}$ in $\mathcal{G}$ model mathematical operations (e.g. TotalCount models "$total = count_1 + count_2$"). Their union should isomorphically model the equation E.




```
% Guess a single type for each entity.
1 { entity_type(Entity, Type): concrete_type(Type) } 1 ← entity(Entity).
instanceof(Entity, Type1) ← entity_type(Entity, Type), subtype(Type, Type1).
% Guess a relation and an assignment of typed arguments for each fact.
1 { fact_relation(Fact, Rel): relation(Rel) } 1 ← fact(Fact).
1 { fact_argument(Fact, K, Entity): instanceof(Entity, Type) } 1 ←
        fact_relation(Fact, Rel),
        K = 1..@arity(Rel), argument_type(Rel, K, Type).

% Deduce whether a logical graph G models an equation E. Its math
% relations should form a subgraph whose shape is isomorphic to E.
models(Eq, Fact) ← fact_relation(Fact, Rel), math_skeleton(Rel, Skel),
                   shape_matches(Eq, Fact, Skel).
shape_matches(Eq, Fact, Skel) ← … % Deduce inductively from arguments.

% Forbid solutions that do not model the required equation.
← equation(Eq), #count { Fact: matches(Eq, Fact) } == 0.
```

$\mathcal{G}$:



Plausible logical situations ≠ Engaging story narrative!

3. Discourse Tropes

Discourse tropes are literary constraints on the logical graph, mined from typical narratives in a setting. Each fact $f \in \mathcal{F}$ must be driven either by *math,* or by some *discourse trope.*

Definition. A *discourse trope* $\mathcal{D}$ is a constraint on $\mathcal{G}$ of form: $\forall \vec{x} \subset \mathcal{E}\colon\ [\Phi(\vec{x}) \Rightarrow \exists \vec{y} \subset \mathcal{E}\colon \Psi(\vec{x}, \vec{y})]$

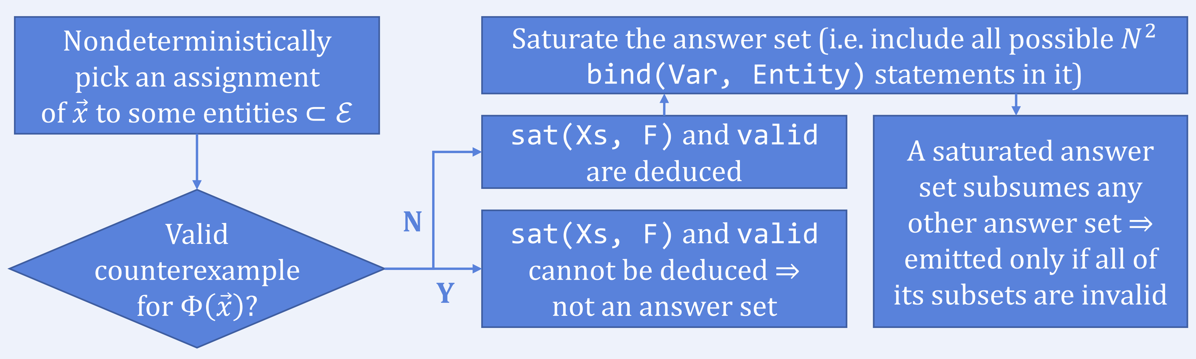
Example. "A warrior slays a monster only if the monster has treasures": $\forall w, m \in \mathcal{E}\colon\ [\text{Slays}(w, m) \Rightarrow \exists t \in \mathcal{E}\colon \text{Owns}(m, t)]$

$\exists \mathcal{G}\colon \text{Models}(\mathcal{G}, Req) \wedge \cdots$ ⇒ 3QBF formula! ∉ NP

Solving discourse trope validation in ASP:

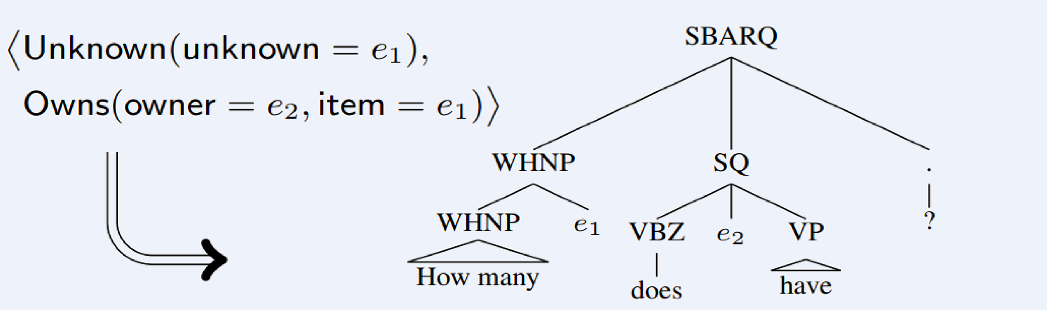
1. Eliminate innermost ∃ with skolemization.
2. Apply *saturation technique* [7] to enforce ∃∀ validation:

```
% Example discourse trope: ∀a,b ∈ ℰ: Owns(a,b) ∨ Acquires(a,b).
discourse(forall(a,b), premise(or(owns(a,b), acquires(a,b)))).

% Assign each formal variable V ∈ {a,b} to some entity e ∈ ℰ.
bind(Var, Entity): entity(Entity) ← var(Var).
sat(Xs, F) ← … % Deduced if Φ(x⃗) holds under the current assignment x⃗.

valid ← discourse(Xs, F), sat(Xs, F).
bind(Var, Entity) ← valid, var(Var), entity(Entity).  % Saturate.
← not valid.
```



Natural language generation

4. Sentence ordering
   a. Convert each fact $f \in \mathcal{F}$ to a sentence using a database of *primitive templates.*



   b. Temporal and causal connectives $\mathcal{C}$ define a *partial ordering* between sentences ⇒ Build a linear narrative.

5. Reference resolution

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?

- *Non-repetitive references:* "describe the entity with different features every time"
- *Unambiguous references:* "differ from all other previously mentioned entities"

⟹ ∀ reference: find a minimal unambiguous subset of its descriptive features.[8]

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?

(or a Wizardry variation)

Professor Smaug assigns Ellie to make a luck potion. She had to spend 9 hours first reading the recipe in the textbook. She spends several hours brewing 11 portions of it. The potion has to be brewed for 3 hours per portion. How many hours did Ellie spend in total?

Evaluation

Goal: evaluate *generation techniques* by assessing *comprehensibility* and *solubility* of the word problems' content.

Study design:

- Sample 25 generated word problems with sufficient variability.
- Match with 25 equivalent Singapore Math[9] word problems.
- Conduct 2 Amazon Mechanical Turk studies (1000 subjects each):

A. Evaluate the word problem text with respect to given questions on a *forced-choice Likert scale* ("−", "∓", "±", "+", mapped to 1-4).

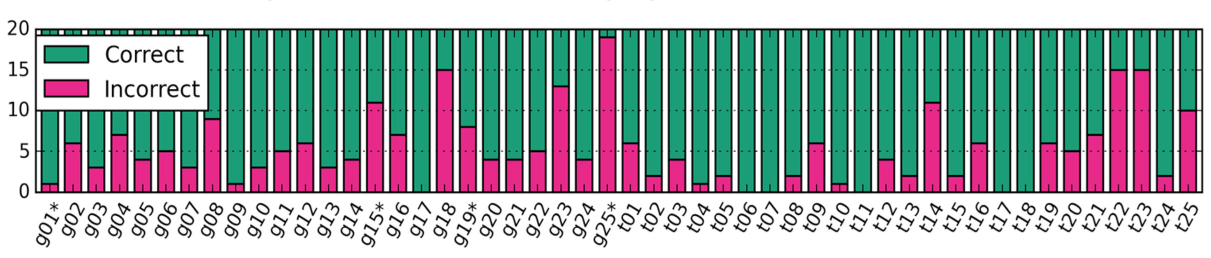
Q1: How comprehensible is the problem? How well did you understand the plot?
Q2: How logical and natural is the sentence order?
Q3: When the problem refers to an actor (e.g. with a pronoun or a name), is it clear who is being mentioned?
Q4: Do the numbers in the problem fit its story (e.g. it would not make sense for a knight to be 5 years old)?

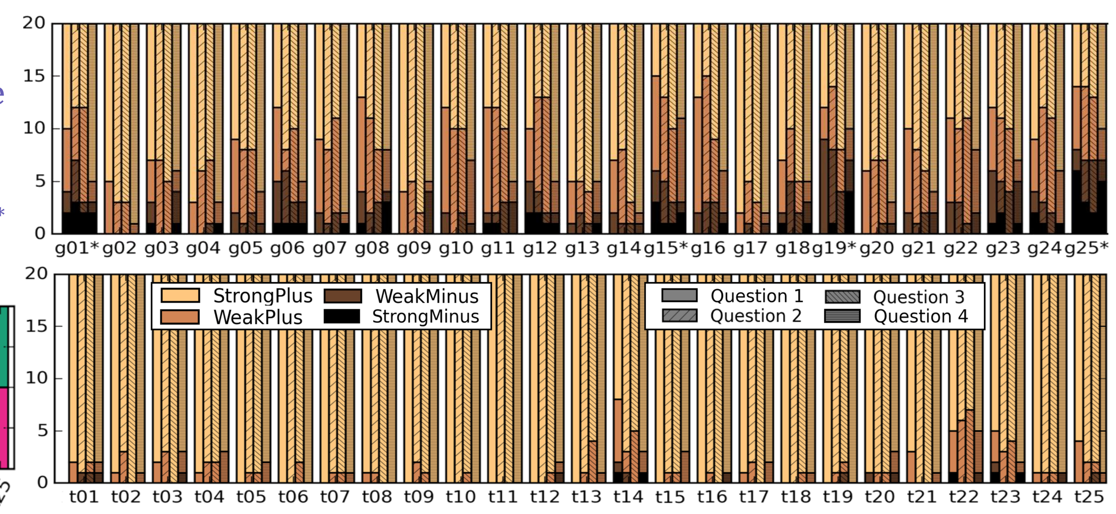
B. Solve the word problem. Correctness and solving time are recorded.

Findings

- ✓ Generated problems are rated equally or slightly less comprehensible than the textbook problems ($\chi^2 = 193.52,\ p < 0.001,\ V = 0.44$).
- ✓ Generated problems are generally comprehensible ($\mu \approx 3.45 - 3.65$).
- ✓ Solubility of generated problems is indistinguishable from textbook.*

* After removing 4 outliers with unclear language.





References

1. Lieven Verschaffel. Using retelling data to study elementary school children's representations and solutions of compare problems. *Journal for Research in Mathematics Education*, pages 141–165, 1994.
2. Denise Dellarosa Cummins, Walter Kintsch, Kurt Reusser, and Rhonda Weimer. The role of understanding in solving word problems. *Cognitive psychology*, 20(4):405–438, 1988.
3. Robin F Schumacher and Lynn S Fuchs. Does understanding relational terminology mediate effects of intervention on compare word problems? *Journal of experimental child psychology*, 111(4):607–628, 2012.
4. Jacque Ensign. *Linking life experiences to classroom math*. PhD thesis, University of Virginia, 1996.
5. Janis M Hart. The effect of personalized word problems. *Teaching Children Mathematics*, 2(8):504–505, 1996.
6. Martin Gebser, Roland Kaminski, Benjamin Kaufmann, and Torsten Schaub. Answer set solving in practice. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(3):1–238, 2012.
7. Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner. Answer set programming: A primer. In *Reasoning Web. Semantic Technologies for Information Systems*, pages 40–110. Springer, 2009.
8. Emiel Krahmer and Kees Van Deemter. Computational generation of referring expressions: A survey. *Computational Linguistics*, 38(1):173–218, 2012.
9. Frank Schaffer Publications. *Singapore Math 70 Must-Know Word Problems, Level 3 Grade 4*. Carson-Dellosa Publishing, LLC, 2009.