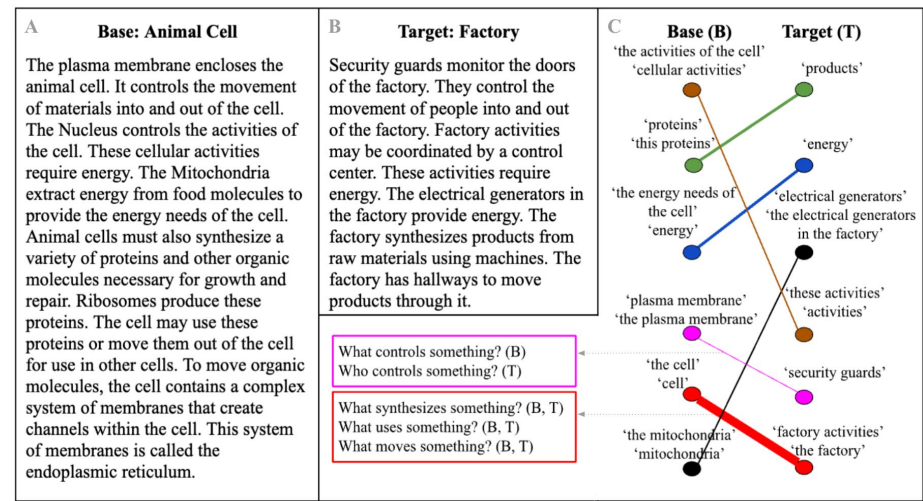


# 1) Analogical Reasoning over Natural Language Paragraphs (2022-2023)



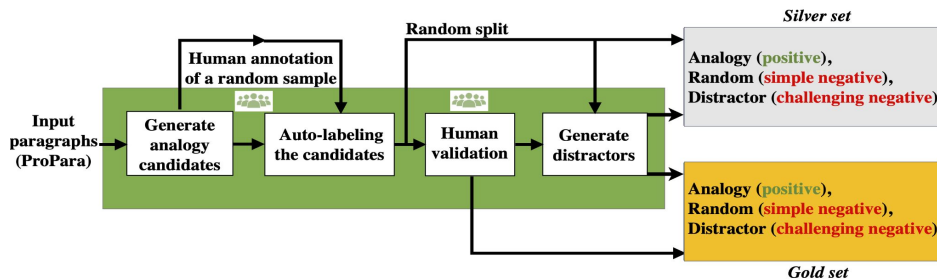
## Life is a Circus and We are the Clowns: Automatically Finding Analogies between Situations and Processes

Oren Sultan, Dafna Shahaf



## ParallelPARC: A Scalable Pipeline for Generating Natural-Language Analogies

Oren Sultan, Yonatan Bitton, Ron Yosef, Dafna Shahaf

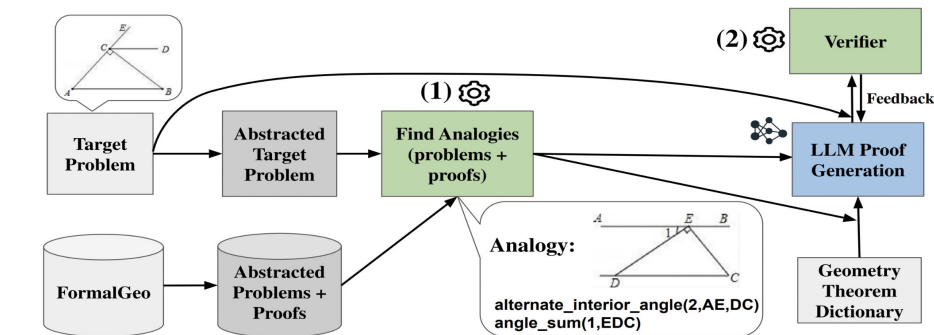


Base	Target	Similar Relations
<p><b>Title:</b> How does a solar panel work?</p> <p><b>Domain:</b> Engineering</p> <p><b>Paragraph:</b> solar energy <i>powers</i> an electric current within a solar panel. The <b>photovoltaic cells</b> within the panel <i>convert</i> the energy from the sun into electricity. The <b>electrical wires</b> then <i>spread</i> this <b>power</b> throughout the panel. The <b>electric current</b> is then used to <i>power</i> whatever the panel is connected to.</p>	<p><b>Title:</b> How does photosynthesis occur?</p> <p><b>Domain:</b> Natural Science</p> <p><b>Paragraph:</b> Photosynthesis occurs when <b>sunlight</b> <i>powers</i> chemical reactions within the <b>chloroplasts</b> of a plant. The <b>chloroplasts</b> are able to <i>transform</i> the energy from the <b>sunlight</b> into usable energy for the <b>plant</b>. This energy is then used to produce <b>nutrients</b> for the plant, which are then <i>distributed</i> throughout the <b>plant</b>.</p>	<p>(solar energy, <i>powers</i>, electric current) (sunlight, <i>powers</i>, chemical reactions)</p> <p>(photovoltaic cells, <i>convert</i>, energy) (chloroplasts, <i>transform</i>, energy)</p> <p>(electrical wires, <i>spread</i>, power) (plants, <i>distribute</i>, nutrients)</p>

## 2) Combining LLMs with tools – A neuro-symbolic approach (2024-2025)

### Towards Reliable Proof Generation with LLMs: A Neuro-Symbolic Approach

Oren Sultan, Eitan Stern, Dafna Shahaf



**Problem:**

**Description:** " $\angle ABC = 40^\circ$ ,  $CD \parallel AB$ ,  $BC$  is perpendicular to  $AC$ . Find the measure of  $\angle ECD$ "

**Construction:**  
 $Shape(CA, AB, BC)$ ,  $Shape(EC, CD)$  ...

**Construction (extended):**  
 $Shape(AB, BC, CA)$ ,  $Shape(BC, CA, AB)$  ...

**Conditions:**  
 $Equal(MeasureOfAngle(ABC), 40)$ ,  
 $ParallelBetweenLine(CD, AB)$  ...

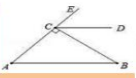
**Goal:**  
 $Value(MeasureOfAngle(ECD))$

**Answer:** 50

**Proof:**  
 1) `parallel_property_alternate_interior_angle(1, CD, AB)`  
 2) `angle_addition(1, ECD, DCB)`  
 3) `adjacent_complementary_angle(1, ECB, BCA)`

**Theorem Dictionary (GDL):**

```
parallel_property_alternate_interior_angle(AB, CD):
{
  1: {
    premises:
    ParallelBetweenLine(AB, CD) & Line(AD),
    conclusions: Equal(MeasureOfAngle(BAD), ...)
  },
  2: { ... },
  ...
},
angle_addition(ABC, CBD): {
  1: { ... },
  ...
}
```

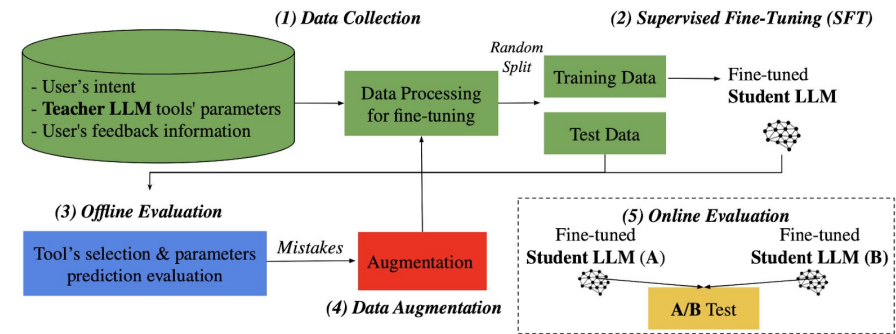


# 2) Combining LLMs with tools – A neuro-symbolic approach (2024)



## Visual Editing with LLM-based Tool Chaining: An Efficient Distillation Approach for Real-Time Applications

Oren Sultan, Alex Khasin, Guy Shiran, Asnat Greenstein-Messica, Dafna Shahaf



### Adjust

```
{
  "exposure": 0,
  "contrast": 10,
  "brightness": 10,
  "highlights": 20,
  "shadows": -10,
  "saturation": 15,
  "vibrance": 15,
  "temperature": 30,
  "tint": 10,
  "hue": 0,
  "bloom": 0,
  "sharpen": 0,
  "structure": 0,
  "linearOffset": 0
}
```

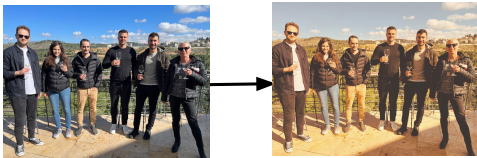
### Selective adjust

```
{
  "red": {"saturation": 20, "luminance": 10},
  "orange": {"saturation": 30, "luminance": 20},
  "yellow": {"saturation": 40, "luminance": 30},
  "green": {"saturation": -20, "luminance": 0},
  "cyan": {"saturation": -20, "luminance": 0},
  "blue": {"saturation": 0, "luminance": 0}
}
```

### Filter

```
{
  "name": "faded_HighNoon",
  "intensity": 40
}
```

### "Golden hour"



# 3) Code World Models (CWM) and the Halting Problem (2025)



**CWM: An Open-Weights LLM for Research on Code Generation with World Models**  
Meta FAIR CodeGen Team, Oren Sultan

Meta FAIR CodeGen Team

< trace_context_start >				
<pre>def count(s, t):     n = 0     for c in s:         n += int(c == t)     return n  count("strawberry", "r") # &lt;&lt; START_OF_TRACE</pre>				
< frame_sep >				
< call_sep >	{ "s": "'strawberry'", "t": "'r'" }	< action_sep >	def count(s, t):	
< frame_sep >				
< line_sep >	{ "s": "..", "t": ".." }	< action_sep >	n = 0	
< frame_sep >				
< line_sep >	{ "s": "..", "t": "..", "n": "0" }	< action_sep >	for c in s:	
< frame_sep >				
< line_sep >	{ "s": "..", "t": "..", "n": "..", "c": "'s'" }	< action_sep >	n += int(c == t)	
...				
< frame_sep >				
< return_sep >	< action_sep >	return n	< arg_sep >	"3"
< frame_sep >				



**LLMs versus the Halting Problem: Revisiting Program Termination Prediction**  
Oren Sultan, Jordi Armengol-Estapé, Pascal Kesseli, Julien Vanegue, Dafna Shahaf, Peter O'Hearn, Yossi Adi

Meta FAIR CodeGen Team

