Exploring the **Learnability** *of* **Program Synthesizers** *by* **Novice Programmers**

Dhanya Jayagopal **Justin Lubin** **Sarah E. Chasins**

EPIC Advance, Fall 2022

Domain experts

(Law, medicine, science, …)

No time to learn to code!

Program synthesis

Automatic generation of code that satisfies a user-provided **specification**

Research question

What aspects of program synthesizers contribute to and detract from their learnability by novice programmers?

- ➡ Qualitative methods!
- ➡ Observe + interview novice programmers working with synthesizers
  - ➥ We provided previously-released synthesizers + tasks
- ➡ Thematic analysis

Results

Tool characteristics

User misconceptions

Why are these important?

Tool characteristics

Understand
learnability implications of ***tool characteristics***
⇓
Make ***empirically-supported design decisions***

User misconceptions

Understand
user misconceptions
⇓
Make systems to ***proactively combat them***

Results

Tool characteristics

User misconceptions

Results

Tool characteristics

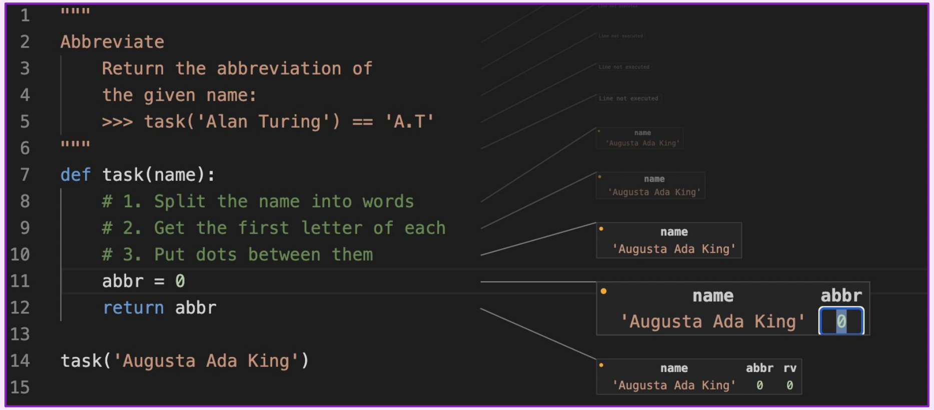
User misconceptions

$$\begin{tikzcd}
\text{Voluntary Specification} \arrow[r, leftrightarrow] & \text{Incidental Specification} \\
\text{User-Triggered Initiation} \arrow[r, leftrightarrow] & \text{Triggerless Initiation} \\
\text{User-Triggered Result Communication} \arrow[r, leftrightarrow] & \text{Triggerless Result Communication}
\end{tikzcd}$$

$$\begin{tikzcd}
\text{Voluntary Specification} \arrow[r, leftrightarrow] & \text{Incidental Specification} \\
\text{User-Triggered Initiation} \arrow[r, leftrightarrow] & \text{Triggerless Initiation} \\
\text{User-Triggered Result Communication} \arrow[r, leftrightarrow] & \text{Triggerless Result Communication}
\end{tikzcd}$$

Voluntary specification

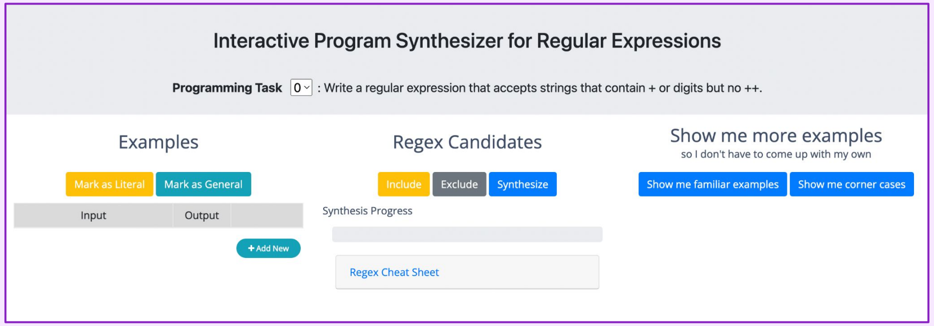
*(**New process** to create specification)*



```
"""
Abbreviate
    Return the abbreviation of
    the given name:
    >>> task('Alan Turing') == 'A.T'
"""
def task(name):
    # 1. Split the name into words
    # 2. Get the first letter of each
    # 3. Put dots between them
    abbr = 0
    return abbr

task('Augusta Ada King')
```

```
name
'Augusta Ada King'

name
'Augusta Ada King'

name
'Augusta Ada King'

name                abbr
'Augusta Ada King'  0

name                abbr  rv
'Augusta Ada King'  0     0
```

"So I'm not allowed to type my own code, I have to do it this way?"



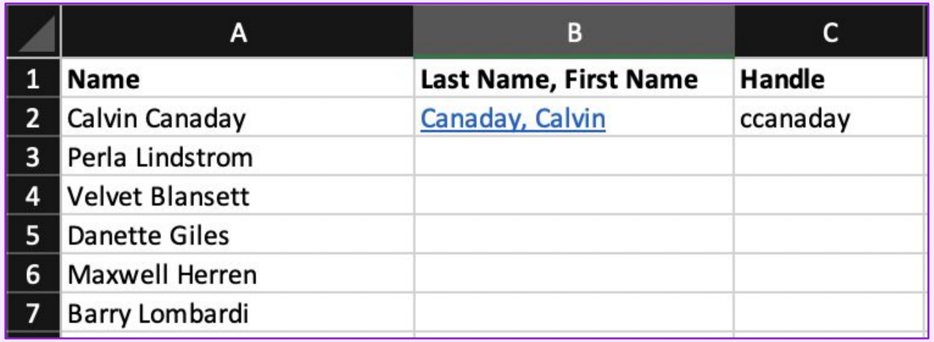
Interactive Program Synthesizer for Regular Expressions

Programming Task 0 : Write a regular expression that accepts strings that contain + or digits but no ++.

"Is there anywhere I can type the regex directly then?"

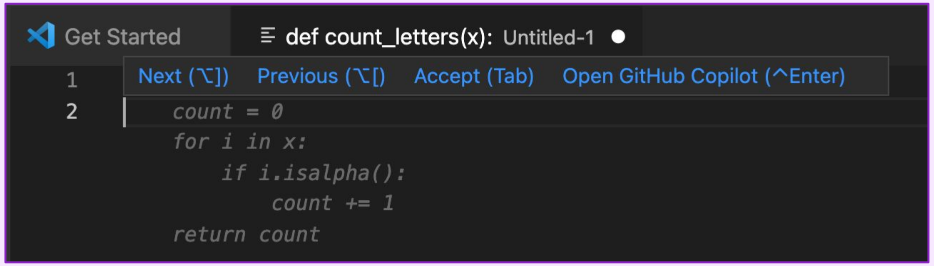
Incidental specification

*(**Existing process** to create specification)*



	A	B	C
1	**Name**	**Last Name, First Name**	**Handle**
2	Calvin Canaday	Canaday, Calvin	ccanaday
3	Perla Lindstrom		
4	Velvet Blansett		
5	Danette Giles		
6	Maxwell Herren		
7	Barry Lombardi		

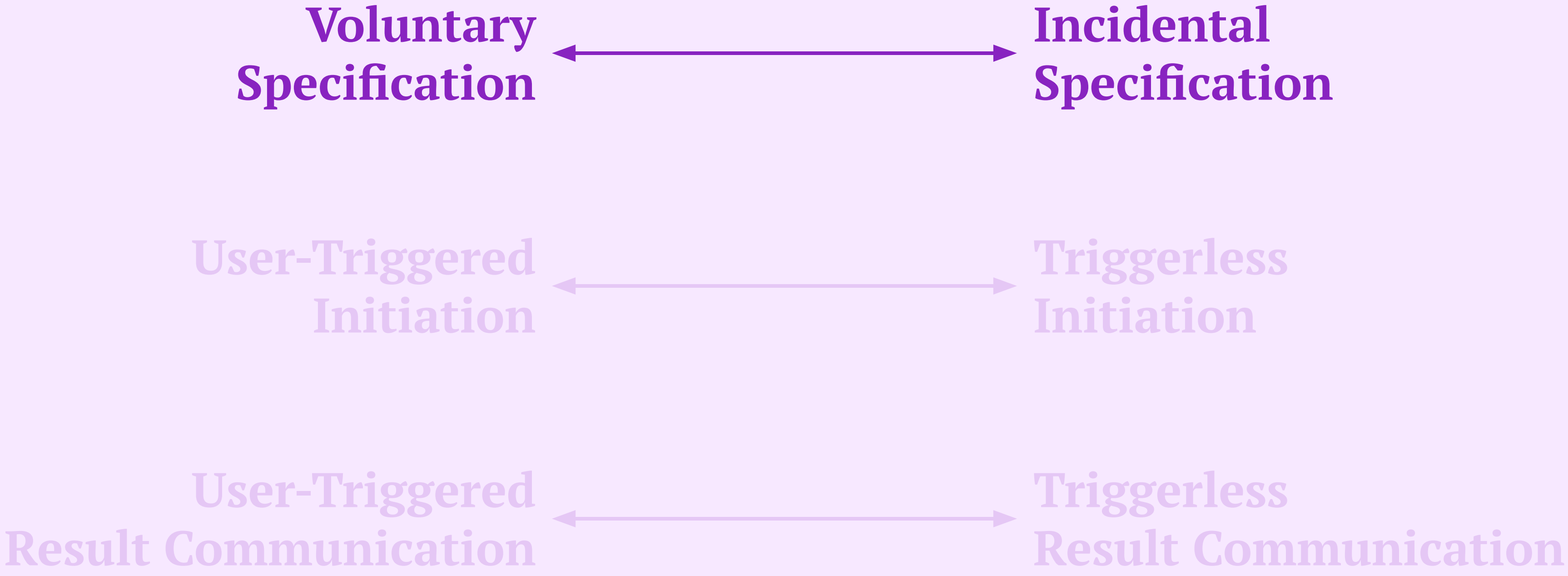
"Oh! That is very convenient. I didn't have to type anything except for the first cell."



Get Started
def count_letters(x): Untitled-1
Next (⌥]) Previous (⌥[) Accept (Tab) Open GitHub Copilot (^Enter)

```
    count = 0
    for i in x:
        if i.isalpha():
            count += 1
    return count
```

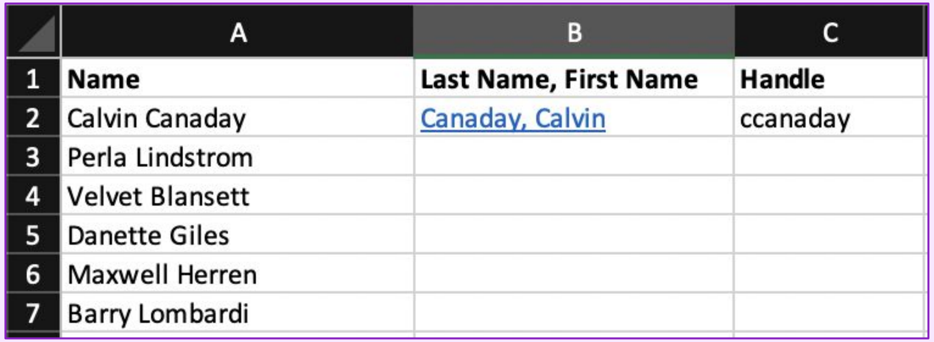
"Oh cool! Okay, so I just have to write code normally."


Voluntary Specification
Incidental Specification
User-Triggered Initiation
Triggerless Initiation
User-Triggered Result Communication
Triggerless Result Communication

$$\begin{tikzcd}
\text{Voluntary Specification} \arrow[r, leftrightarrow] & \text{Incidental Specification} \\
\text{User-Triggered Initiation} \arrow[r, leftrightarrow] & \text{Triggerless Initiation} \\
\text{User-Triggered Result Communication} \arrow[r, leftrightarrow] & \text{Triggerless Result Communication}
\end{tikzcd}$$

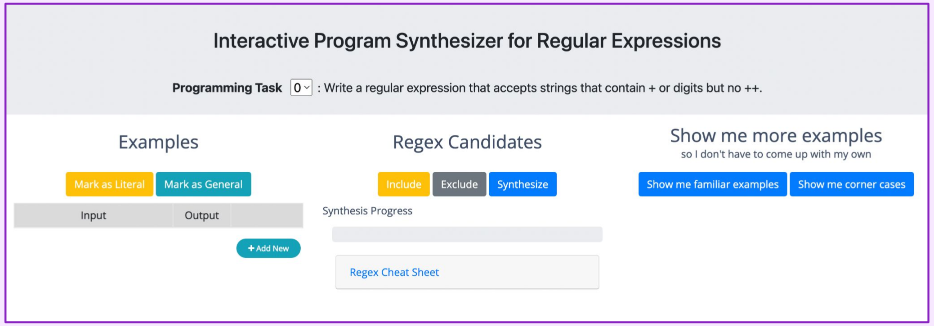
User-triggered initiation

*(**User** decides when to run synthesis)*



	A	B	C
1	**Name**	**Last Name, First Name**	**Handle**
2	Calvin Canaday	Canaday, Calvin	ccanaday
3	Perla Lindstrom		
4	Velvet Blansett		
5	Danette Giles		
6	Maxwell Herren		
7	Barry Lombardi		

Incidental specification, but still user-triggered initiation



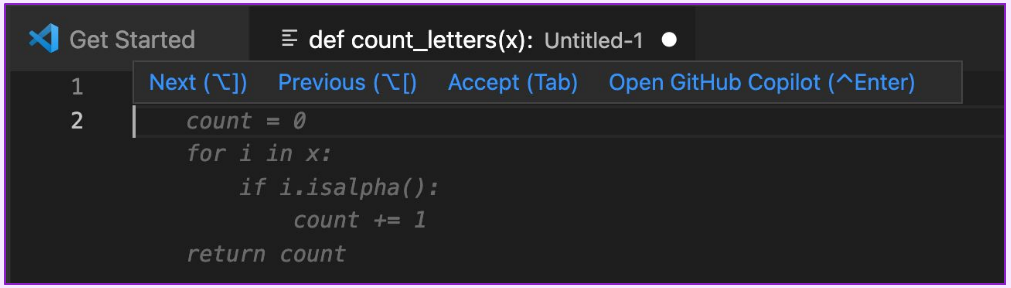
Interactive Program Synthesizer for Regular Expressions

Programming Task 0 : Write a regular expression that accepts strings that contain + or digits but no ++.

"I wasn't sure when to stop adding examples because I thought I had to add one for every possible input."

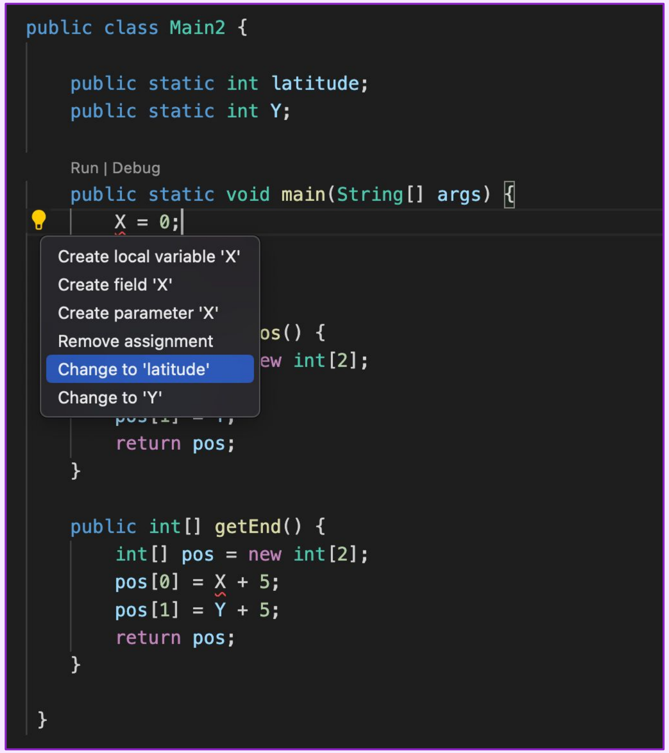
Triggerless initiation

*(**Synthesizer** decides when to run synthesis)*



Get Started
def count_letters(x): Untitled-1
Next (⌥])
Previous (⌥[)
Accept (Tab)
Open GitHub Copilot (^Enter)

```
count = 0
for i in x:
    if i.isalpha():
        count += 1
return count
```



```
public class Main2 {

    public static int latitude;
    public static int Y;

    Run | Debug
    public static void main(String[] args) {
        X = 0;



                            os() {
                            ew int[2];

        return pos;
    }

    public int[] getEnd() {
        int[] pos = new int[2];
        pos[0] = X + 5;
        pos[1] = Y + 5;
        return pos;
    }

}
```

➡ Completely circumvents needing to know how much information to provide!

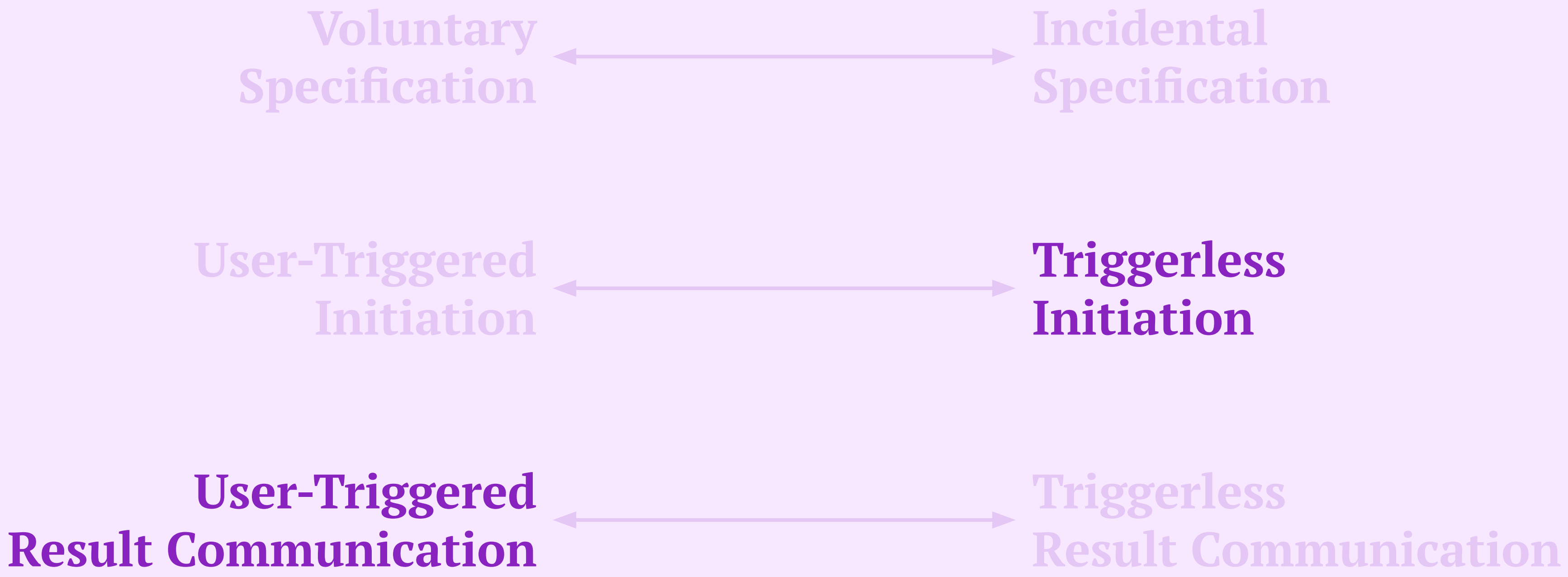
➡ … But sometimes fully automatic is unpredictable

$$\begin{tikzcd}
\text{Voluntary Specification} \arrow[r, leftrightarrow] & \text{Incidental Specification} \\
\text{User-Triggered Initiation} \arrow[r, leftrightarrow] & \text{Triggerless Initiation} \\
\text{User-Triggered Result Communication} \arrow[r, leftrightarrow] & \text{Triggerless Result Communication}
\end{tikzcd}$$

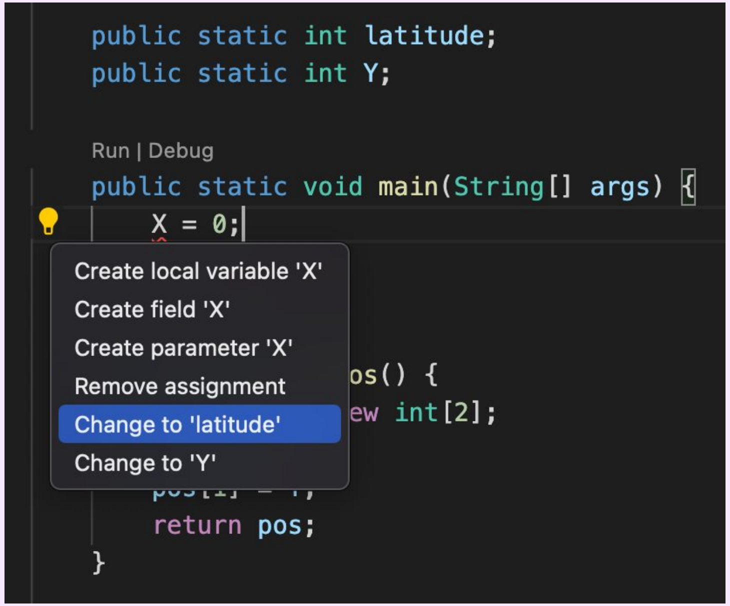
$$\begin{tikzcd}
\text{Voluntary Specification} \arrow[r, leftrightarrow] & \text{Incidental Specification} \\
\text{User-Triggered Initiation} \arrow[r, leftrightarrow] & \text{Triggerless Initiation} \\
\text{User-Triggered Result Communication} \arrow[r, leftrightarrow] & \text{Triggerless Result Communication}
\end{tikzcd}$$

Voluntary Specification	↔	Incidental Specification
User-Triggered Initiation	↔	Triggerless Initiation
User-Triggered Result Communication	↔	Triggerless Result Communication

Voluntary Specification	↔	Incidental Specification
User-Triggered Initiation	↔	**Triggerless Initiation**
User-Triggered Result Communication	↔	**Triggerless Result Communication**


Voluntary Specification
Incidental Specification
User-Triggered Initiation
Triggerless Initiation
User-Triggered Result Communication
Triggerless Result Communication

Triggerless initiation & User-triggered result communication


```
    public static int latitude;
    public static int Y;

    Run | Debug
    public static void main(String[] args) {
        X = 0;
```

```
                        os() {
                        ew int[2];

        return pos;
    }
```

“I didn’t really know when to click on it, because I didn't know how it could help.”

Voluntary Specification ↔ Incidental Specification

User-Triggered Initiation ↔ Triggerless Initiation

User-Triggered Result Communication ↔ Triggerless Result Communication

Voluntary Specification ⟷ **Incidental Specification**

User-Triggered Initiation ⟷ **Triggerless Initiation**

User-Triggered Result Communication ⟷ **Triggerless Result Communication**

Voluntary Specification	↔	**Incidental Specification**
User-Triggered Initiation	↔	**Triggerless Initiation**
User-Triggered Result Communication	↔	**Triggerless Result Communication**

Voluntary
Speci

Incidental
ation

Specification size:

Not as important!

User-Tr
In

ess
n

User-Tr
Result Communication

ess
Result Communication

Results

Tool characteristics

User misconceptions

Results

Tool characteristics

User misconceptions

Incorrectly believing the synthesizer made progress

Incorrectly believing the synthesizer did *not* make progress

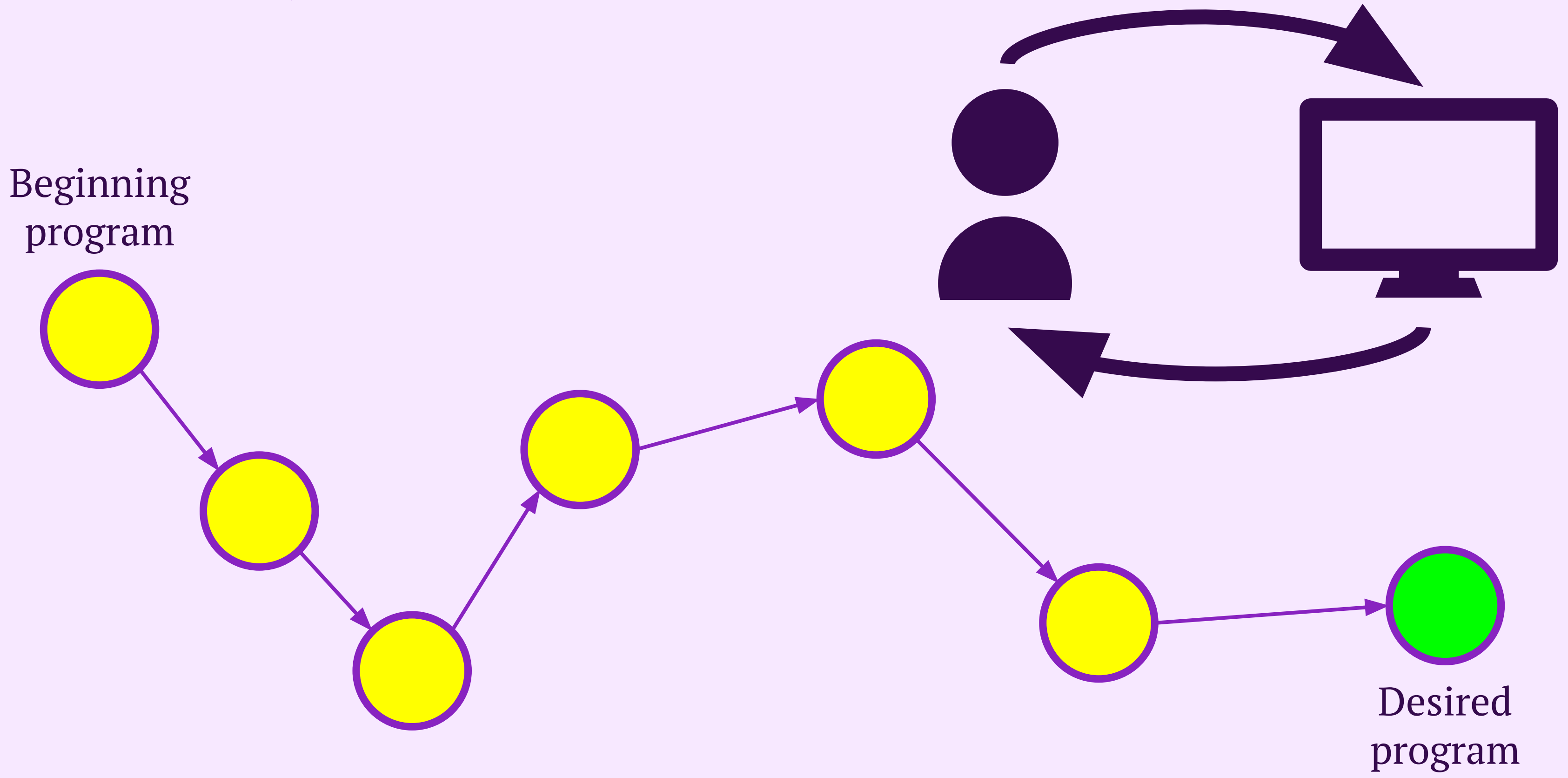
What makes a good specification?

Incorrectly believing the synthesizer made progress

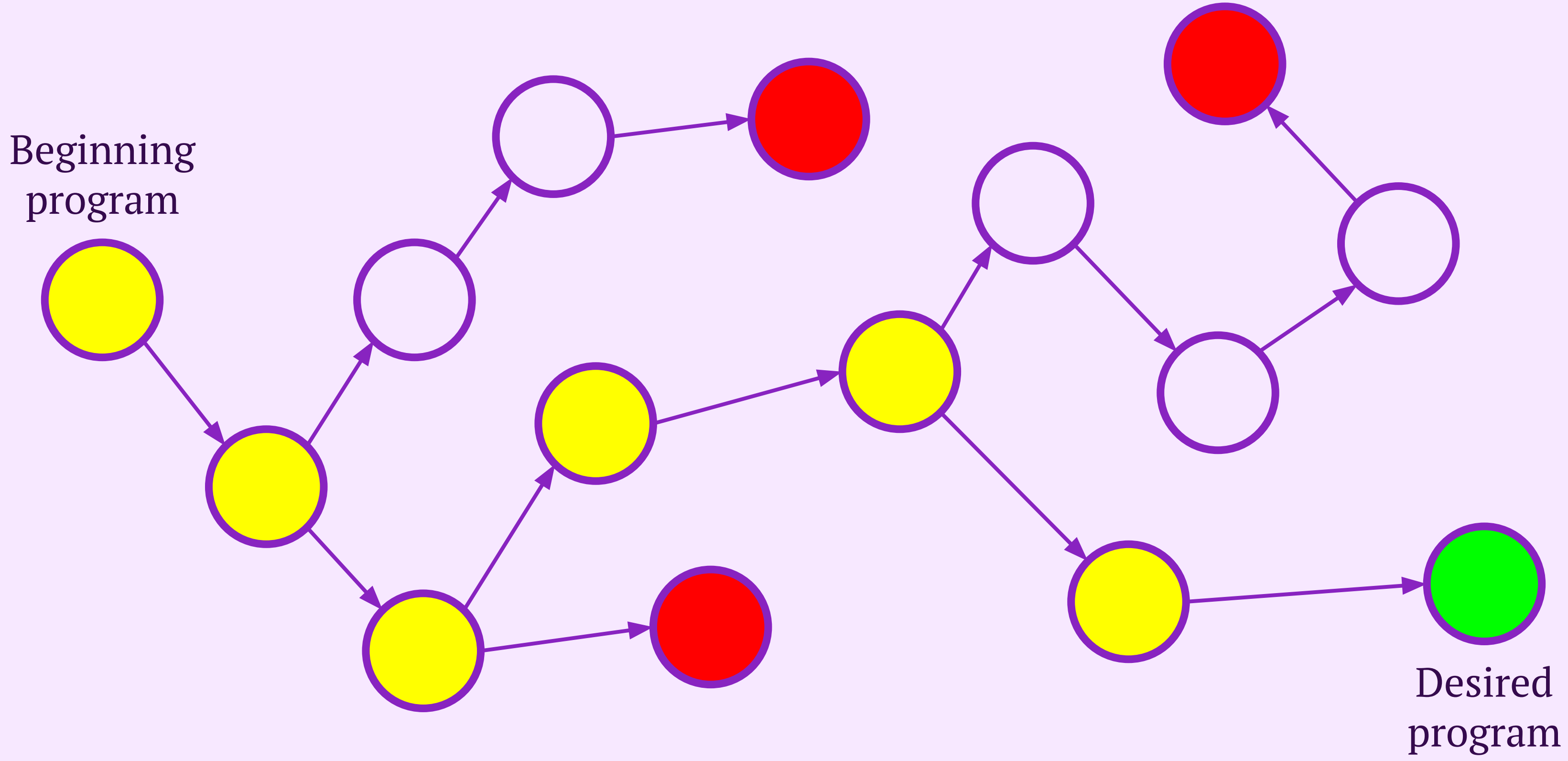
Incorrectly believing the synthesizer did *not* make progress

What makes a good specification?

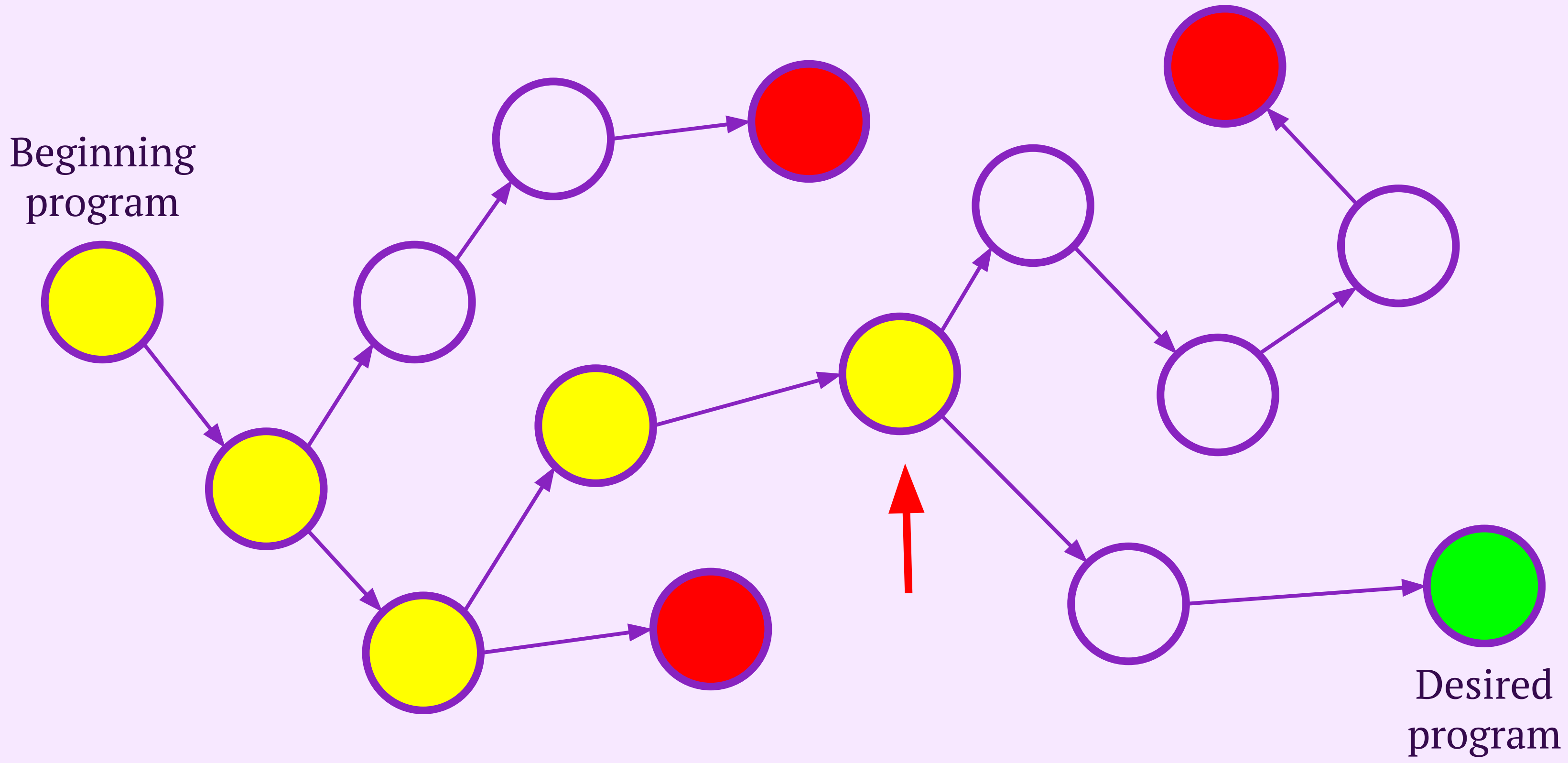
Incorrectly believing progress



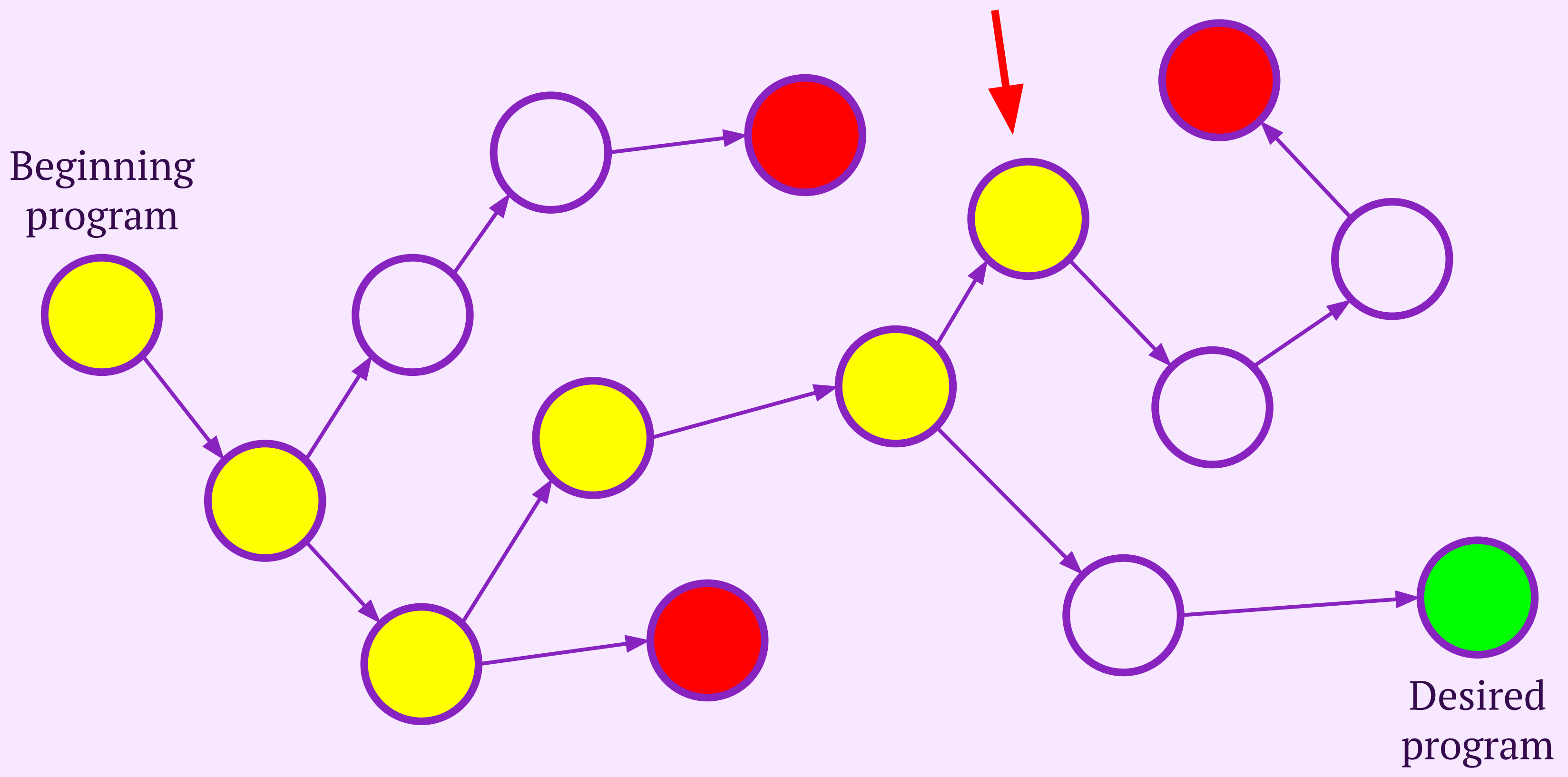
Incorrectly believing progress



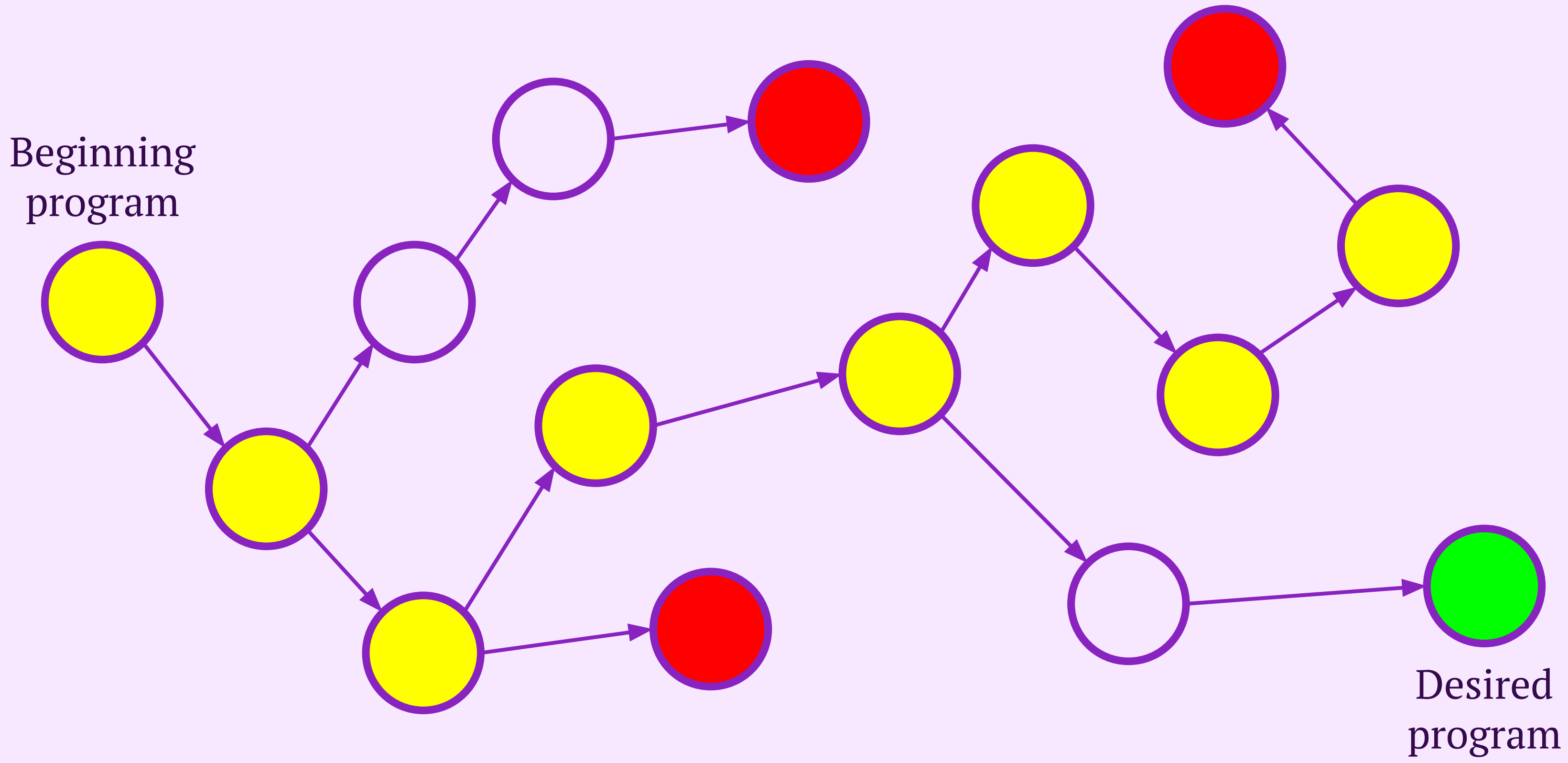
Incorrectly believing progress



Incorrectly believing progress



Incorrectly believing progress

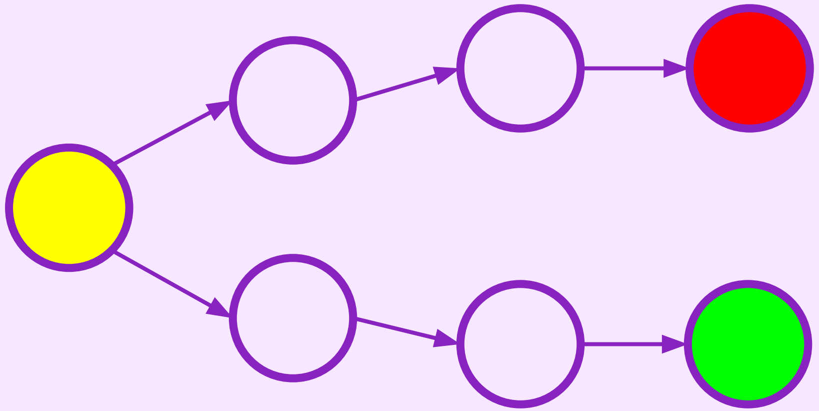


Incorrectly believing progress

- ➡ **Task:** Reverse a given string
- ➡ **Participant's first step:** Split string into space-separated words
- ➡ Synthesis succeeds!
- ➡ … But now what?

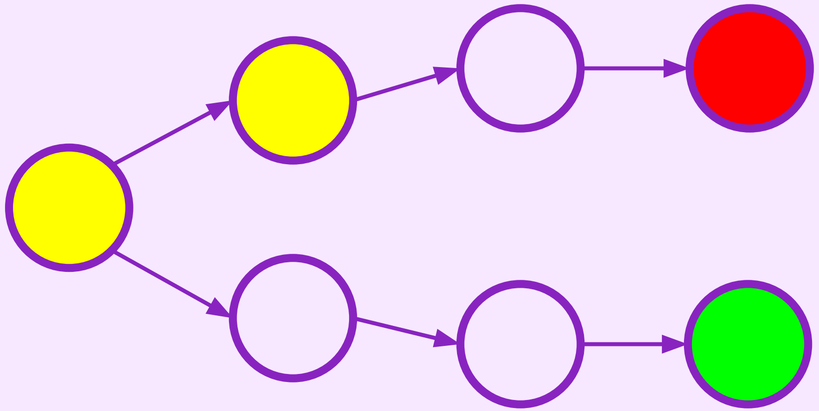
Incorrectly believing progress

- **Task:** Reverse a given string
- **Participant's first step:** Split string into space-separated words
- Synthesis succeeds!
- … But now what?



Incorrectly believing progress

- **Task:** Reverse a given string
- **Participant's first step:** Split string into space-separated words
- Synthesis succeeds!
- … But now what?



Incorrectly believing the synthesizer made progress

Incorrectly believing the synthesizer did *not* make progress

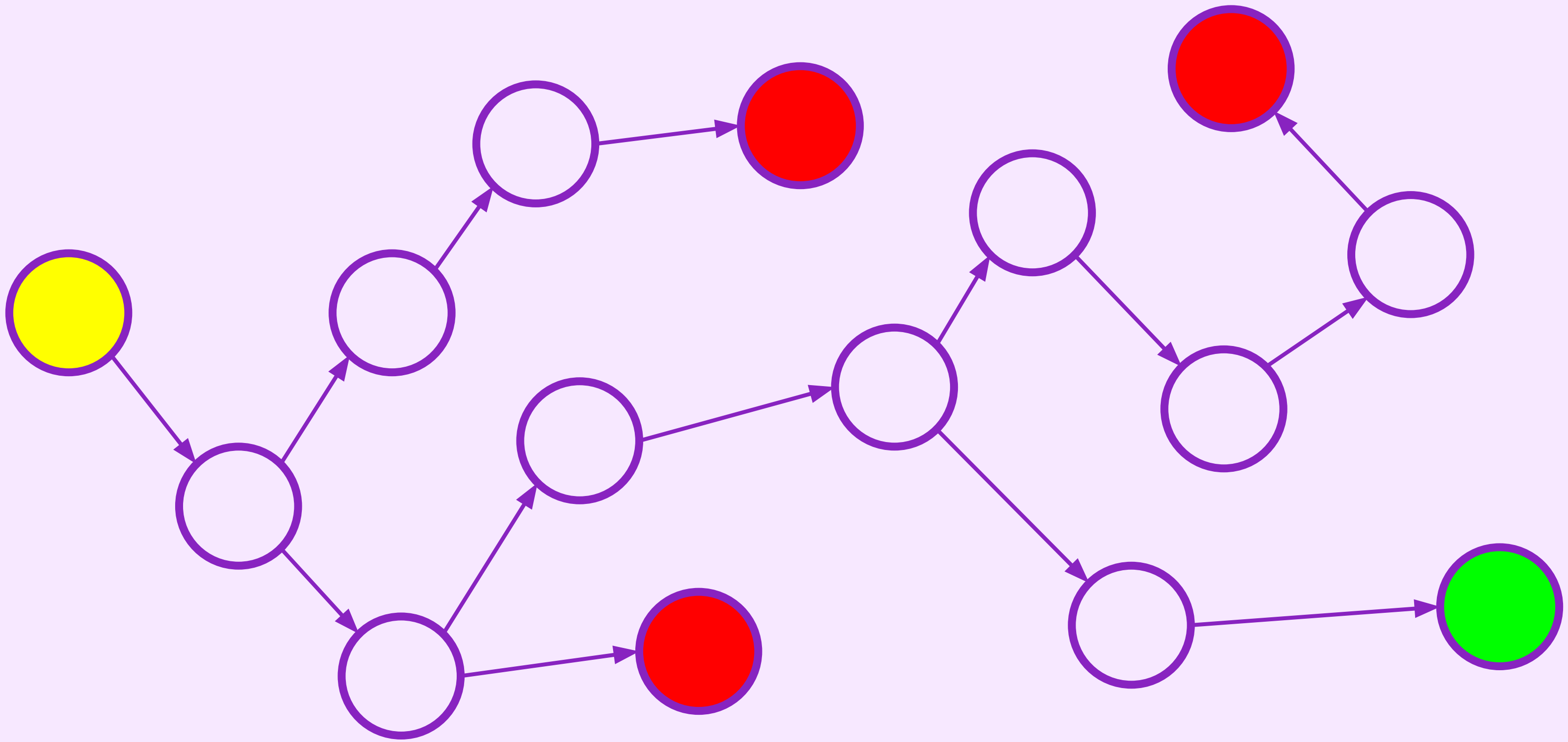
What makes a good specification?

Incorrectly believing the synthesizer made progress

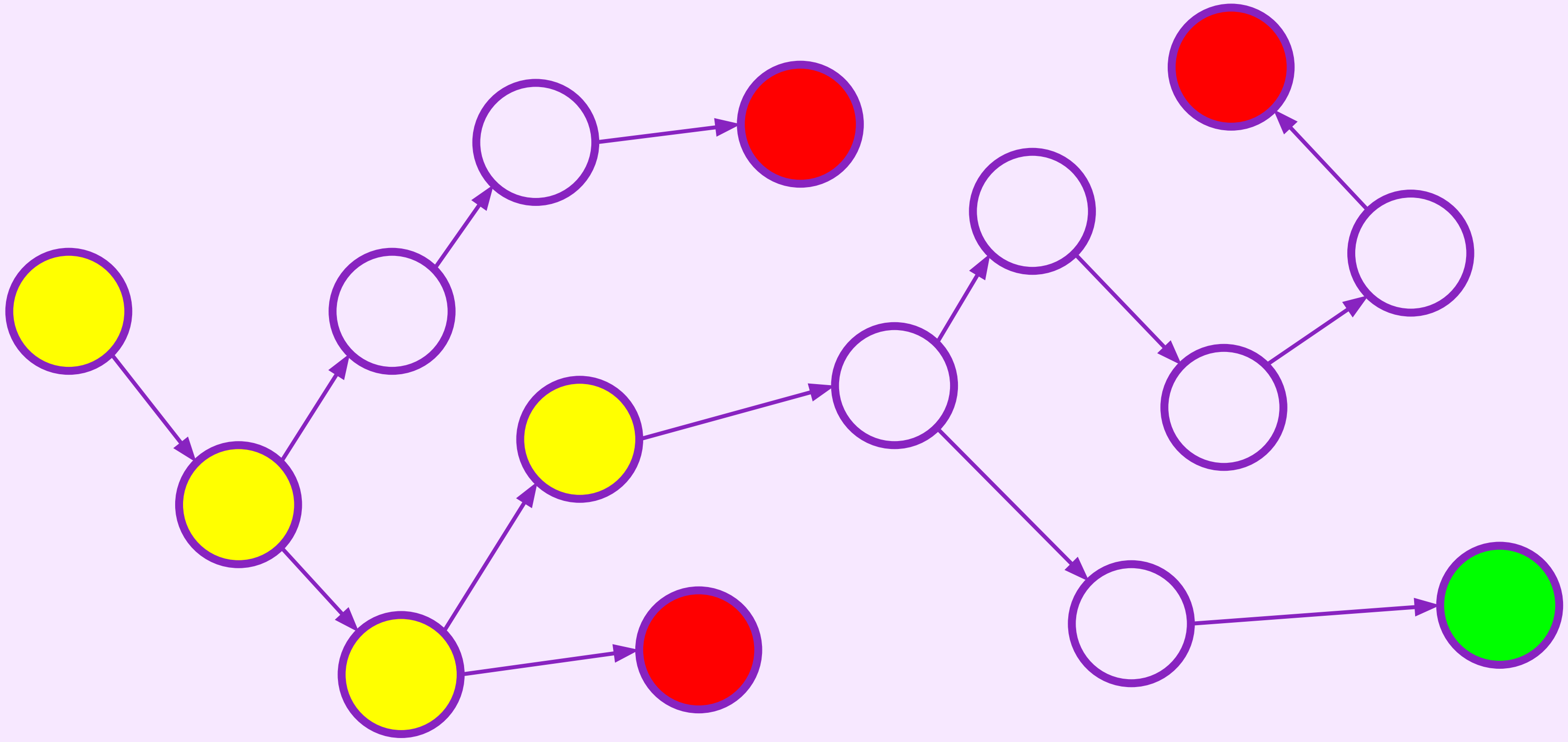
Incorrectly believing the synthesizer did *<u>not</u>* make progress

What makes a good specification?

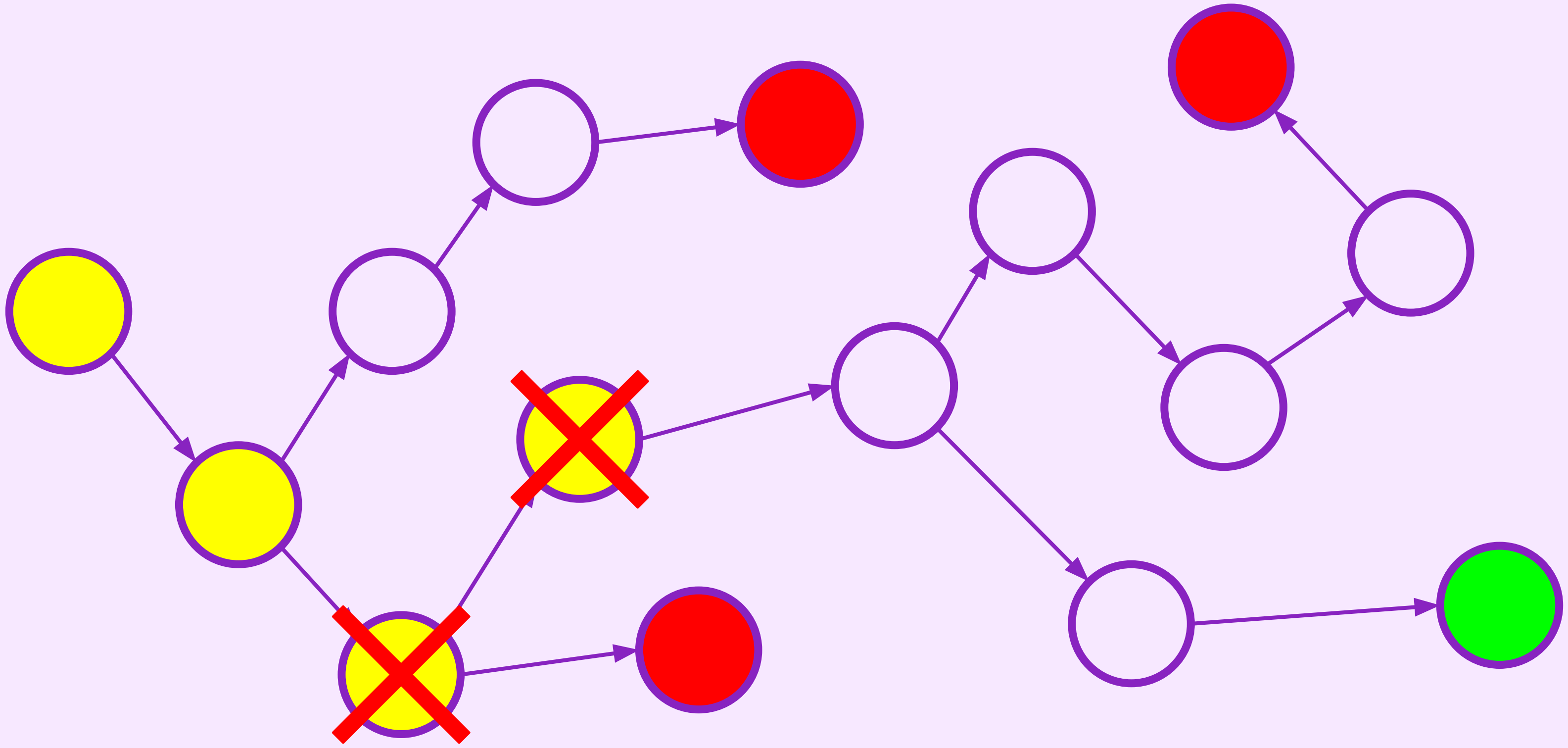
Incorrectly believing *no* progress



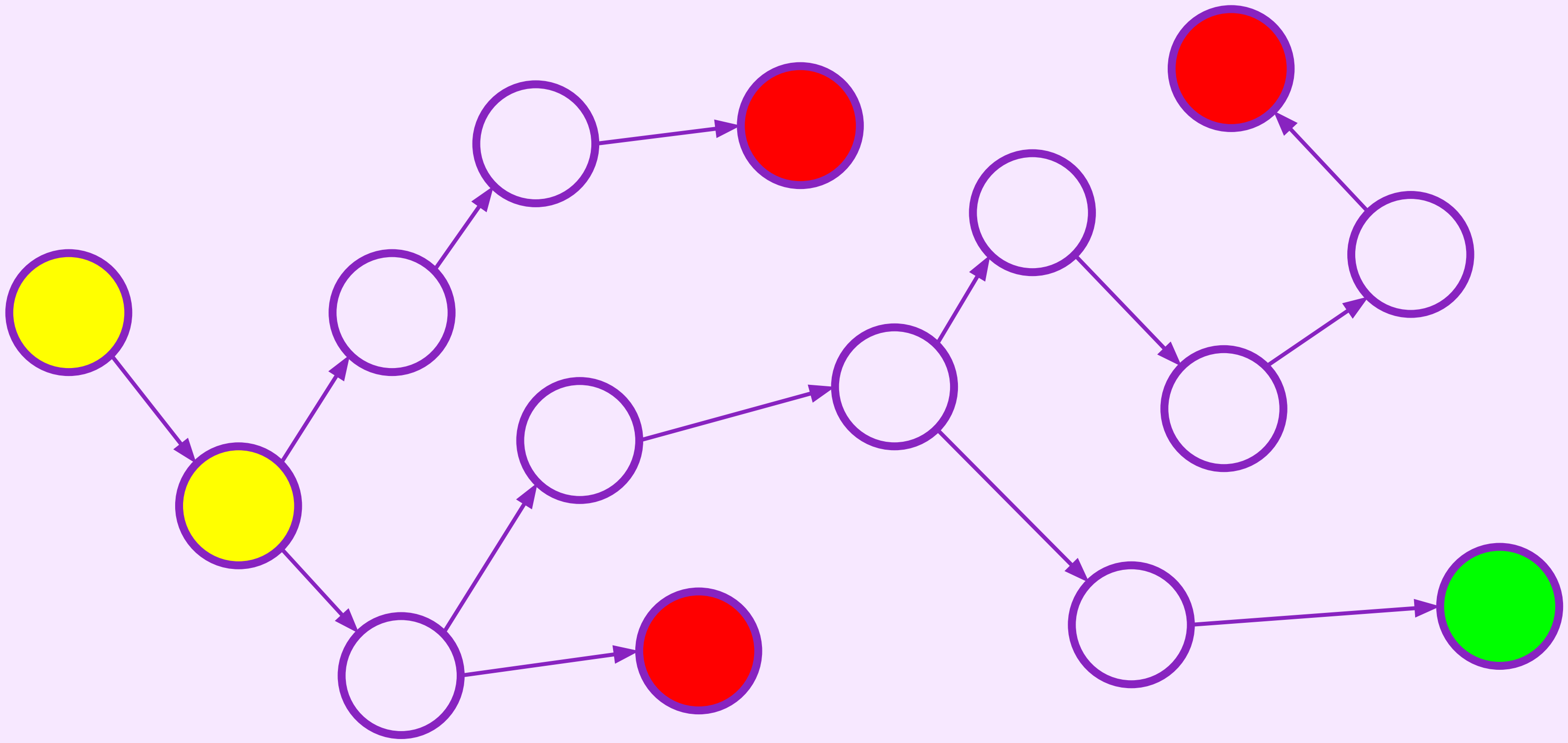
Incorrectly believing *no* progress



Incorrectly believing *no* progress



Incorrectly believing *no* progress

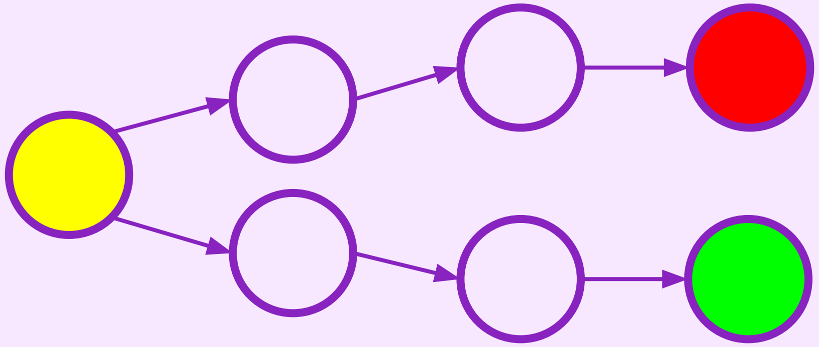


Incorrectly believing *no* progress

- **Task:** Create regex to match + or digits but no ++
- **Participant sees candidate regex** that rejects ++ but accepts letters
- Participant rejects candidate
- … But just needed to slightly refine output

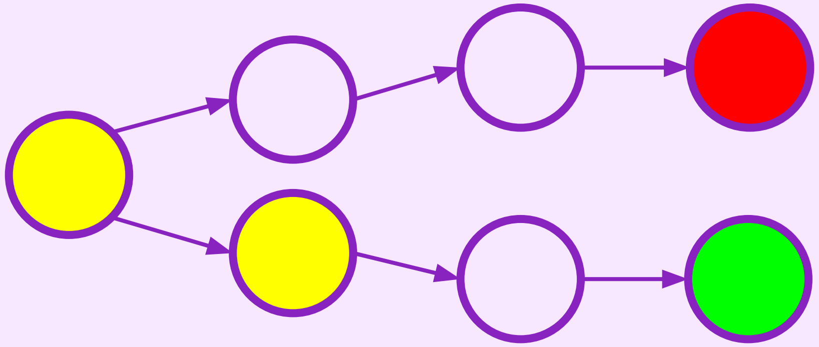
Incorrectly believing *no* progress

- ➡ **Task:** Create regex to match + or digits but no ++
- ➡ **Participant sees candidate regex** that rejects ++ but accepts letters
- ➡ Participant rejects candidate
- ➡ … But just needed to slightly refine output



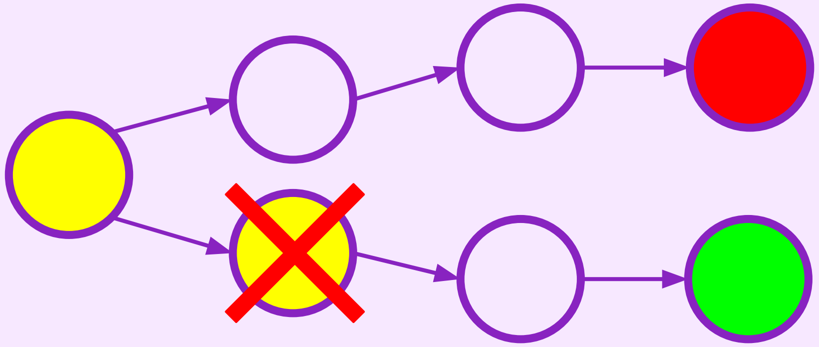
Incorrectly believing *no* progress

- **Task:** Create regex to match + or digits but no ++
- **Participant sees candidate regex** that rejects ++ but accepts letters
- Participant rejects candidate
- … But just needed to slightly refine output



Incorrectly believing *no* progress

- ➡ **Task:** Create regex to match + or digits but no ++
- ➡ **Participant sees candidate regex** that rejects ++ but accepts letters
- ➡ Participant rejects candidate
- ➡ … But just needed to slightly refine output



Incorrectly believing the synthesizer made progress

Incorrectly believing the synthesizer did ***not*** make progress

What makes a good specification?

Incorrectly believing the synthesizer made progress

Incorrectly believing the synthesizer did *not* make progress

What makes a good specification?

What makes a good specification?

- **Triggerless tools**
  - "It feels like the light bulb pops up randomly."
- **Example-based specifications**
  - "Am I missing any cases? I feel like I covered all of the edge cases. Do I need to add a different example for every letter?"
- **Lack of feedback upon failure**
  - "Is there a way to check why it failed?"

Incorrectly believing the synthesizer made progress

Incorrectly believing the synthesizer did *not* make progress

What makes a good specification?

Incorrectly believing the synthesizer made progress

Incorrectly believing the synthesizer did *<u>not</u>* make progress

What makes a good specification?

More in the paper!

Results

Tool characteristics

User misconceptions

Exploring the **Learnability** *of* **Program Synthesizers** *by* **Novice Programmers**

Voluntary Specification $\longleftrightarrow$ **Incidental Specification**

User-Triggered Initiation $\longleftrightarrow$ **Triggerless Initiation**

User-Triggered Result Communication $\longleftrightarrow$ **Triggerless Result Communication**

Incorrectly believing the synthesizer made progress

Incorrectly believing the synthesizer did *not* make progress

What makes a good specification?

Thanks to: Sarah and Dhanya, our anonymous participants, and **you**!