Oriented Metrics for Bottom-Up Enumerative Synthesis

Roland Meyer and Jakob Tepe

TU Braunschweig, Germany

Goal:

Unify and **generalize**

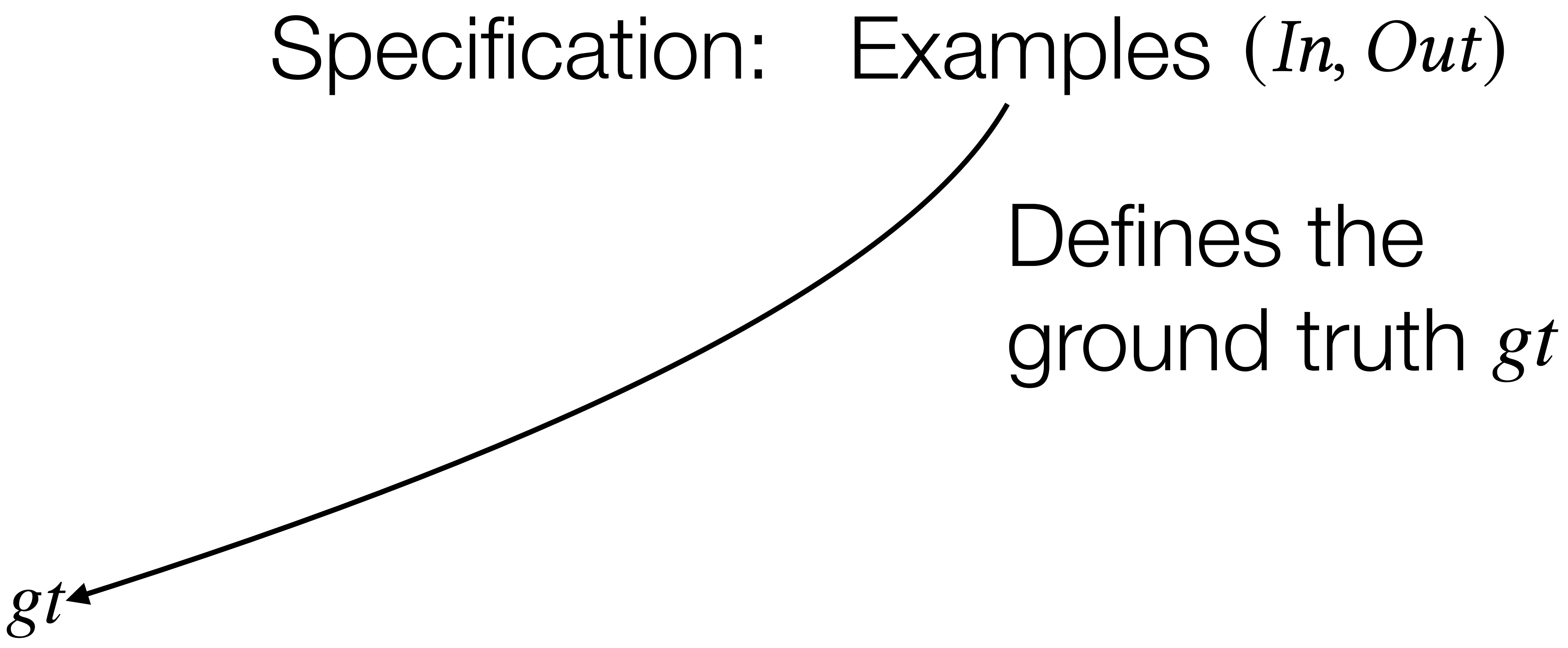
the existing approaches to bottom-up enumerative synthesis

Syntax-Guided Synthesis (SyGuS)

Syntax-Guided Synthesis (SyGuS)

Specification: Examples (In, Out)

Syntax-Guided Synthesis (SyGuS)



Specification: Examples (In, Out)

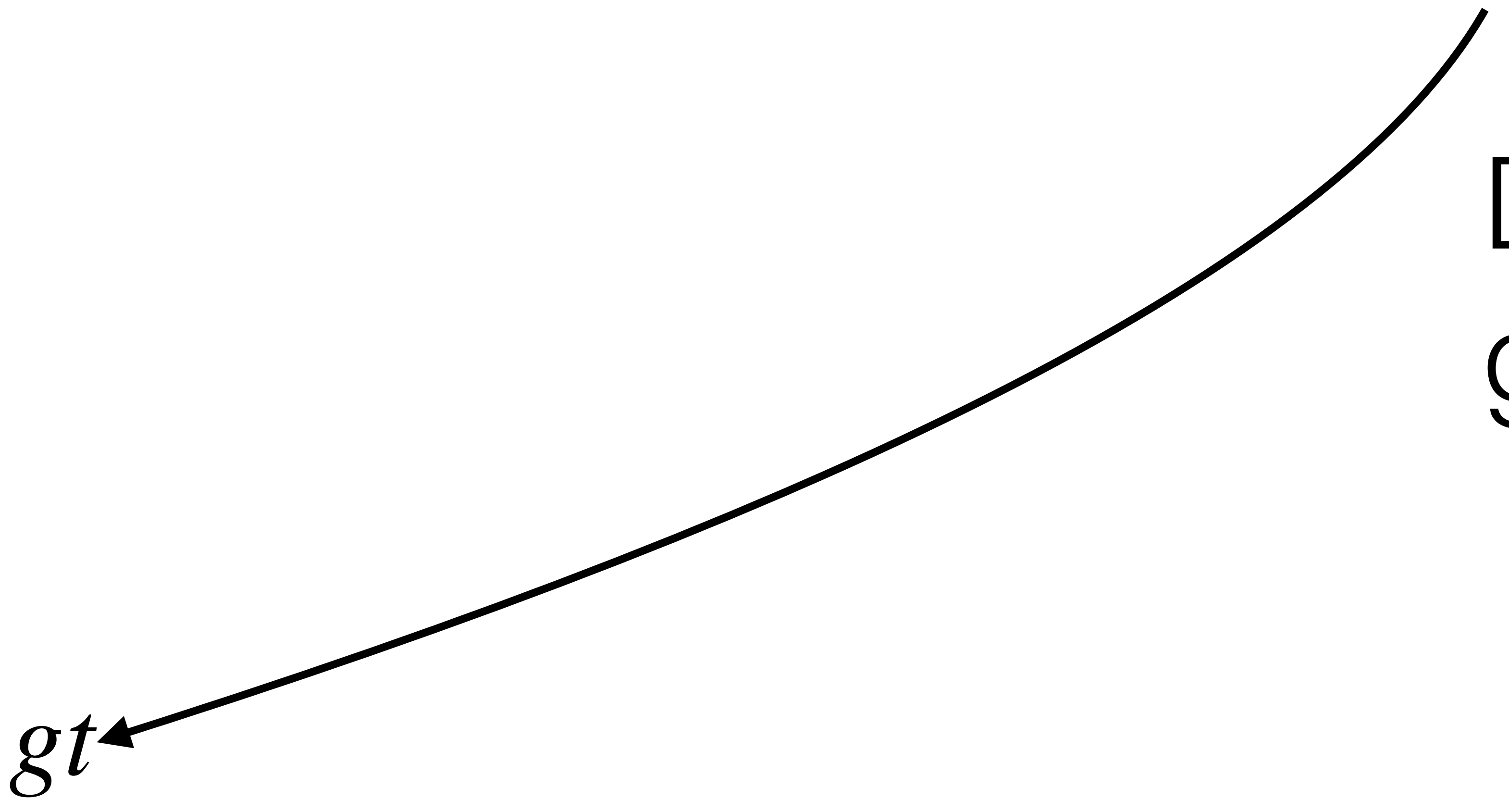
Defines the ground truth gt

gt

Syntax-Guided Synthesis (SyGuS)

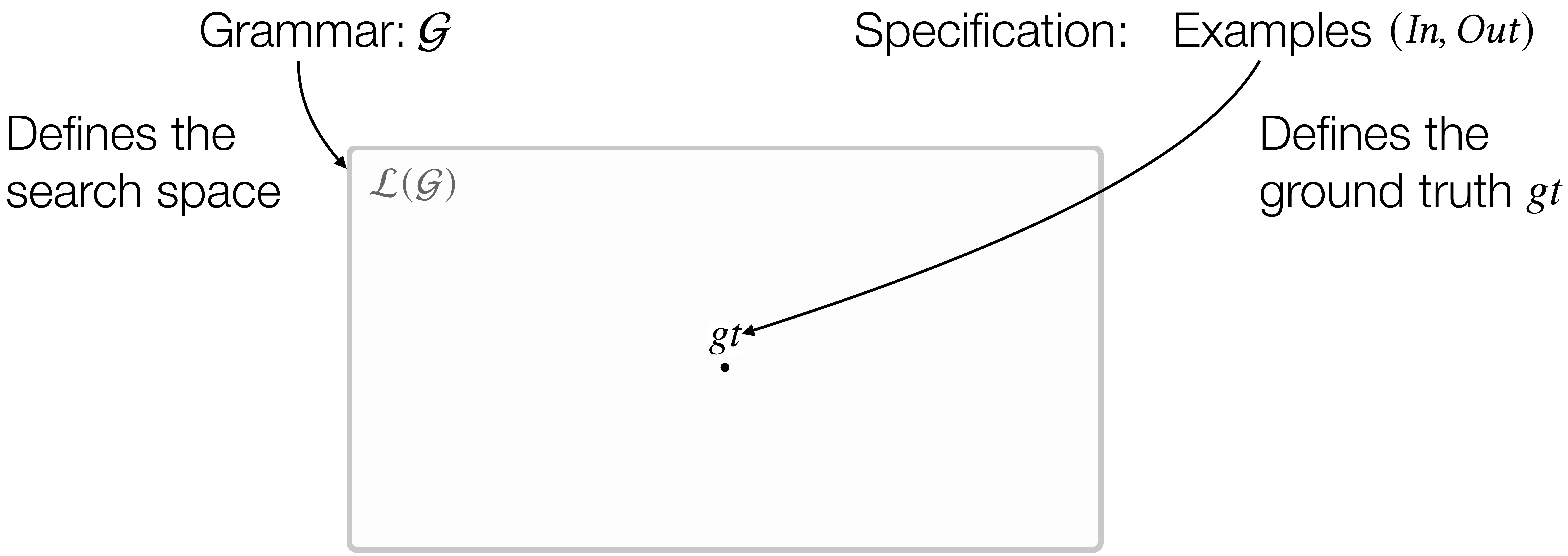
Grammar: $\mathcal{G}$

Specification: Examples (In, Out)



Defines the ground truth gt

Syntax-Guided Synthesis (SyGuS)



Grammar: $\mathcal{G}$

Specification: Examples (In, Out)

Defines the search space

Defines the ground truth gt

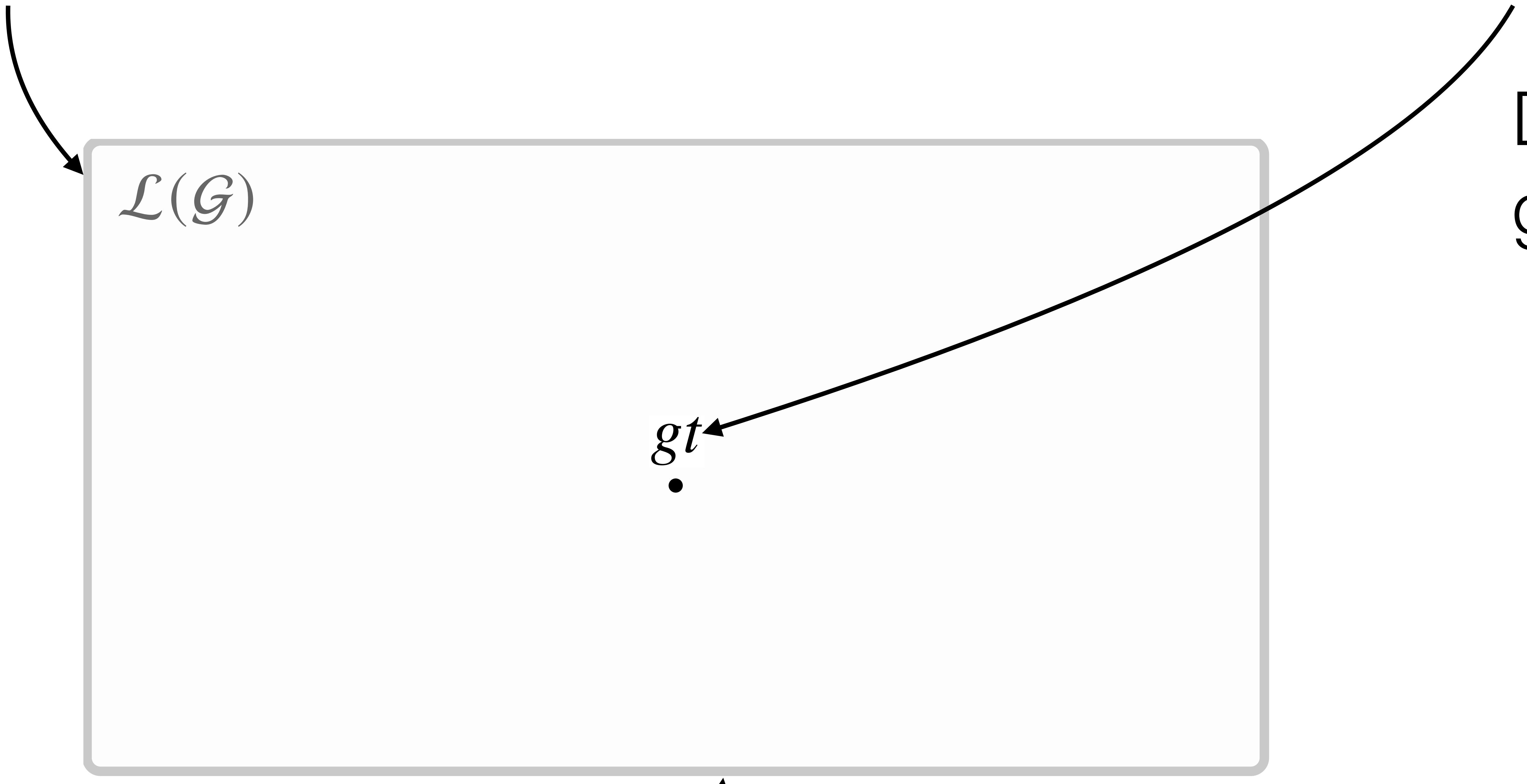
Syntax-Guided Synthesis (SyGuS)

Grammar: $\mathcal{G}$

Specification: Examples (In, Out)

Defines the search space

Defines the ground truth gt



Task: Find program **prog** $\in$ that implements gt

Syntax-Guided Synthesis (SyGuS)

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

Syntax-Guided Synthesis (SyGuS)

$$\mathcal{S} \quad ::= \quad V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S})$$

$$V \quad ::= \quad \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

Syntax-Guided Synthesis (SyGuS)

$$\begin{array}{lll} \mathcal{S} & ::= & V \mid \texttt{replace}(\mathcal{S}, \mathcal{S}, \mathcal{S}) \mid \texttt{concat}(\mathcal{S}, \mathcal{S}) \\ V & ::= & \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"} \end{array}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

prog$(i) = o$

Syntax-Guided Synthesis (SyGuS)

$\mathcal{S}$ ::= V | `replace`$(\mathcal{S}, \mathcal{S}, \mathcal{S})$ | `concat`$(\mathcal{S}, \mathcal{S})$

V ::= `x` | ϵ | `"␣Conf"` | `"␣City"`

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

prog$(i) = o$

Solution: `r`(`r`(`x`, `"␣Conf"`, ϵ), `"␣City"`, ϵ)

Syntax-Guided Synthesis (SyGuS)

$$\begin{array}{lll} \mathcal{S} & ::= & V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S}) \\ V & ::= & \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"} \end{array}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

prog$(i) = o$

Solution: $\texttt{r}(\texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon), \texttt{"␣City"}, \epsilon)$

$\texttt{r}(\texttt{r}(\texttt{"POPL␣Conf"}, \texttt{"␣Conf"}, \epsilon), \texttt{"␣City"}, \epsilon)$

Syntax-Guided Synthesis (SyGuS)

$$\mathcal{S} \quad ::= \quad V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S})$$

$$V \quad ::= \quad \texttt{x} \mid \epsilon \mid \texttt{"\textvisiblespace Conf"} \mid \texttt{"\textvisiblespace City"}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

$$\mathsf{prog}(i) = o$$

Solution: $\texttt{r}(\texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon), \texttt{"\textvisiblespace City"}, \epsilon)$

$$\texttt{r}(\underbrace{\texttt{r}(\texttt{"POPL\textvisiblespace Conf"}, \texttt{"\textvisiblespace Conf"}, \epsilon)}_{}, \texttt{"\textvisiblespace City"}, \epsilon)$$

Syntax-Guided Synthesis (SyGuS)

$$\mathcal{S} \quad ::= \quad V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S})$$

$$V \quad ::= \quad \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

$$\text{prog}(i) = o$$

Solution: $\texttt{r}(\texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon), \texttt{"␣City"}, \epsilon)$

$$\texttt{r}(\underbrace{\texttt{r}(\texttt{"POPL␣Conf"}, \texttt{"␣Conf"}, \epsilon)}_{\texttt{"POPL"}}, \texttt{"␣City"}, \epsilon)$$

Syntax-Guided Synthesis (SyGuS)

$$\begin{array}{lll} \mathcal{S} & ::= & V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S}) \\ V & ::= & \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"} \end{array}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

prog$(i) = o$

Solution: $\texttt{r}(\texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon), \texttt{"␣City"}, \epsilon)$

$$\underbrace{\texttt{r}(\underbrace{\texttt{r}(\texttt{"POPL␣Conf"}, \texttt{"␣Conf"}, \epsilon)}_{\texttt{"POPL"}}, \texttt{"␣City"}, \epsilon)}$$

Syntax-Guided Synthesis (SyGuS)

$$\begin{array}{lcl} \mathcal{S} & ::= & V \mid \texttt{replace}(\mathcal{S}, \mathcal{S}, \mathcal{S}) \mid \texttt{concat}(\mathcal{S}, \mathcal{S}) \\ V & ::= & \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"} \end{array}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

prog$(i) = o$

Solution: $\texttt{r}(\texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon), \texttt{"␣City"}, \epsilon)$

$$\underbrace{\texttt{r}(\underbrace{\texttt{r}(\texttt{"POPL␣Conf"}, \texttt{"␣Conf"}, \epsilon)}_{\texttt{"POPL"}}, \texttt{"␣City"}, \epsilon)}_{\texttt{"POPL"}}$$

Syntax-Guided Synthesis (SyGuS)

$$\mathcal{S} \quad ::= \quad V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S})$$

$$V \quad ::= \quad \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

prog$(i) = o$

Solution: $\texttt{r}(\texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon), \texttt{"␣City"}, \epsilon)$

$$\underbrace{\texttt{r}(\underbrace{\texttt{r}(\texttt{"POPL␣Conf"}, \texttt{"␣Conf"}, \epsilon)}_{\texttt{"POPL"}}, \texttt{"␣City"}, \epsilon)}_{\texttt{"POPL"}}$$

$$\texttt{r}(\texttt{r}(\texttt{"Rennes␣City"}, \texttt{"␣Conf"}, \epsilon), \texttt{"␣City"}, \epsilon)$$

Syntax-Guided Synthesis (SyGuS)

$$\begin{array}{lll} \mathcal{S} & ::= & V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S}) \\ V & ::= & \texttt{x} \mid \epsilon \mid \texttt{"\textvisiblespace Conf"} \mid \texttt{"\textvisiblespace City"} \end{array}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

prog$(i) = o$

Solution: $\texttt{r}(\texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon), \texttt{"\textvisiblespace City"}, \epsilon)$

$$\underbrace{\texttt{r}(\underbrace{\texttt{r}(\texttt{"POPL\textvisiblespace Conf"}, \texttt{"\textvisiblespace Conf"}, \epsilon)}_{\texttt{"POPL"}}, \texttt{"\textvisiblespace City"}, \epsilon)}_{\texttt{"POPL"}}$$

$$\texttt{r}(\underbrace{\texttt{r}(\texttt{"Rennes\textvisiblespace City"}, \texttt{"\textvisiblespace Conf"}, \epsilon)}, \texttt{"\textvisiblespace City"}, \epsilon)$$

Syntax-Guided Synthesis (SyGuS)

$\mathcal{S}$ ::= V | $\texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S})$ | $\texttt{concat}(\mathcal{S},\mathcal{S})$

V ::= x | ϵ | "␣Conf" | "␣City"

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

prog(i) = o

Solution: $\texttt{r}(\texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon), \texttt{"␣City"}, \epsilon)$

$$\underbrace{\texttt{r}(\underbrace{\texttt{r}(\texttt{"POPL␣Conf"}, \texttt{"␣Conf"}, \epsilon)}_{\texttt{"POPL"}}, \texttt{"␣City"}, \epsilon)}_{\texttt{"POPL"}}$$

$$\texttt{r}(\underbrace{\texttt{r}(\texttt{"Rennes␣City"}, \texttt{"␣Conf"}, \epsilon)}_{\texttt{"Rennes␣City"}}, \texttt{"␣City"}, \epsilon)$$

Syntax-Guided Synthesis (SyGuS)

$$\mathcal{S} \quad ::= \quad V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S})$$

$$V \quad ::= \quad \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

$\text{prog}(i) = o$

Solution: $\texttt{r}(\texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon), \texttt{"␣City"}, \epsilon)$

$$\underbrace{\texttt{r}(\underbrace{\texttt{r}(\texttt{"POPL␣Conf"}, \texttt{"␣Conf"}, \epsilon)}_{\texttt{"POPL"}}, \texttt{"␣City"}, \epsilon)}_{\texttt{"POPL"}}$$

$$\underbrace{\texttt{r}(\underbrace{\texttt{r}(\texttt{"Rennes␣City"}, \texttt{"␣Conf"}, \epsilon)}_{\texttt{"Rennes␣City"}}, \texttt{"␣City"}, \epsilon)}$$

Syntax-Guided Synthesis (SyGuS)

$$\begin{array}{lll} \mathcal{S} & ::= & V \mid \texttt{replace}(\mathcal{S}, \mathcal{S}, \mathcal{S}) \mid \texttt{concat}(\mathcal{S}, \mathcal{S}) \\ V & ::= & \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"} \end{array}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

prog$(i) = o$

Solution: $\texttt{r}(\texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon), \texttt{"␣City"}, \epsilon)$

$$\underbrace{\texttt{r}(\underbrace{\texttt{r}(\texttt{"POPL␣Conf"}, \texttt{"␣Conf"}, \epsilon)}_{\texttt{"POPL"}}, \texttt{"␣City"}, \epsilon)}_{\texttt{"POPL"}}$$

$$\underbrace{\texttt{r}(\underbrace{\texttt{r}(\texttt{"Rennes␣City"}, \texttt{"␣Conf"}, \epsilon)}_{\texttt{"Rennes␣City"}}, \texttt{"␣City"}, \epsilon)}_{\texttt{"Rennes"}}$$

Syntax-Guided Synthesis (SyGuS)

$$\mathcal{S} \quad ::= \quad V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S})$$

$$V \quad ::= \quad \texttt{x} \mid \epsilon \mid \texttt{"\textvisiblespace Conf"} \mid \texttt{"\textvisiblespace City"}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

$$\text{prog}(i) = o$$

Solution: $\texttt{r}(\texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon), \texttt{"\textvisiblespace City"}, \epsilon)$

$$\underbrace{\texttt{r}(\underbrace{\texttt{r}(\texttt{"POPL\textvisiblespace Conf"}, \texttt{"\textvisiblespace Conf"}, \epsilon)}_{\texttt{"POPL"}}, \texttt{"\textvisiblespace City"}, \epsilon)}_{\texttt{"POPL"}}$$

$$\underbrace{\texttt{r}(\underbrace{\texttt{r}(\texttt{"Rennes\textvisiblespace City"}, \texttt{"\textvisiblespace Conf"}, \epsilon)}_{\texttt{"Rennes\textvisiblespace City"}}, \texttt{"\textvisiblespace City"}, \epsilon)}_{\texttt{"Rennes"}}$$

Understanding 1: Existing approaches

have a way to **enumerate**

Bottom-Up Enumeration

$$\begin{array}{lll} \mathcal{S} & ::= & V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S}) \\ V & ::= & \texttt{x} \mid \epsilon \mid \texttt{"\textvisiblespace Conf"} \mid \texttt{"\textvisiblespace City"} \end{array}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

$$\begin{array}{lcl} \mathcal{S} & ::= & V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S}) \\ V & ::= & \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"} \end{array}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

Bottom-Up Enumeration

$$P_1 = \{\texttt{x}, \epsilon, \texttt{"␣Conf"}, \texttt{"␣City"}\}$$

$$\begin{aligned}\mathcal{S} &::= V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S})\\ V &::= \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"}\end{aligned}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

Bottom-Up Enumeration

$P_1 = \{\texttt{x}, \epsilon, \texttt{"␣Conf"}, \texttt{"␣City"}\}$

$P_2 = \emptyset$

$$\begin{array}{lcl}\mathcal{S} & ::= & V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S}) \\ V & ::= & \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"}\end{array}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

Bottom-Up Enumeration

$$P_1 = \{\texttt{x}, \epsilon, \texttt{"␣Conf"}, \texttt{"␣City"}\} \qquad\qquad P_2 = \emptyset$$

$$\begin{aligned}P_3 = \{&\texttt{x}.\texttt{x}, \texttt{x}.\epsilon, \texttt{x}.\texttt{"␣Conf"}, \texttt{x}.\texttt{"␣City"}, \epsilon.\texttt{x}, \epsilon.\epsilon, \epsilon.\texttt{"␣Conf"}, \epsilon.\texttt{"␣City"},\\ &\texttt{"␣Conf"}.\texttt{x}, \texttt{"␣Conf"}.\epsilon, \texttt{"␣Conf"}.\texttt{"␣Conf"}, \texttt{"␣City"}.\texttt{"␣City"},\\ &\texttt{"␣Conf"}.\texttt{"␣City"}, \texttt{"␣City"}.\texttt{x}, \texttt{"␣City"}.\epsilon, \texttt{"␣City"}.\texttt{"␣Conf"}\}\end{aligned}$$

$$\mathcal{S} \quad ::= \quad V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S})$$

$$V \quad ::= \quad \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

Bottom-Up Enumeration

$P_1 = \{\texttt{x}, \epsilon, \texttt{"␣Conf"}, \texttt{"␣City"}\}$ $\qquad P_2 = \emptyset$

$$\begin{aligned} P_3 = \{&\texttt{x.x}, \texttt{x.}\epsilon, \texttt{x."␣Conf"}, \texttt{x."␣City"}, \epsilon.\texttt{x}, \epsilon.\epsilon, \epsilon.\texttt{"␣Conf"}, \epsilon.\texttt{"␣City"},\\ &\texttt{"␣Conf".x}, \texttt{"␣Conf".}\epsilon, \texttt{"␣Conf"."␣Conf"}, \texttt{"␣City"."␣City"},\\ &\texttt{"␣Conf"."␣City"}, \texttt{"␣City".x}, \texttt{"␣City".}\epsilon, \texttt{"␣City"."␣Conf"}\} \end{aligned}$$

$$P_4 = \{\texttt{r}(\texttt{x}, \texttt{x}, \texttt{"␣City"}), \texttt{r}(\texttt{x}, \texttt{"␣City"}, \texttt{"␣Conf"}), \texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon), \ldots\}$$

Bottom-Up Enumeration

$$\begin{array}{lcl}\mathcal{S} & ::= & V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S}) \\ V & ::= & \texttt{x} \mid \epsilon \mid \texttt{"\textvisiblespace Conf"} \mid \texttt{"\textvisiblespace City"}\end{array}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

$P_1 = \{\texttt{x}, \epsilon, \texttt{"\textvisiblespace Conf"}, \texttt{"\textvisiblespace City"}\}$ $\qquad$ $P_2 = \emptyset$

$$\begin{aligned}P_3 = \{&\texttt{x}.\texttt{x}, \texttt{x}.\epsilon, \texttt{x}.\texttt{"\textvisiblespace Conf"}, \texttt{x}.\texttt{"\textvisiblespace City"}, \epsilon.\texttt{x}, \epsilon.\epsilon, \epsilon.\texttt{"\textvisiblespace Conf"}, \epsilon.\texttt{"\textvisiblespace City"}, \\ &\texttt{"\textvisiblespace Conf"}.\texttt{x}, \texttt{"\textvisiblespace Conf"}.\epsilon, \texttt{"\textvisiblespace Conf"}.\texttt{"\textvisiblespace Conf"}, \texttt{"\textvisiblespace City"}.\texttt{"\textvisiblespace City"}, \\ &\texttt{"\textvisiblespace Conf"}.\texttt{"\textvisiblespace City"}, \texttt{"\textvisiblespace City"}.\texttt{x}, \texttt{"\textvisiblespace City"}.\epsilon, \texttt{"\textvisiblespace City"}.\texttt{"\textvisiblespace Conf"}\}\end{aligned}$$

$P_4 = \{\texttt{r}(\texttt{x}, \texttt{x}, \texttt{"\textvisiblespace City"}), \texttt{r}(\texttt{x}, \texttt{"\textvisiblespace City"}, \texttt{"\textvisiblespace Conf"}), \texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon), \ldots\}$

$P_5 = \{\ldots\}$

$$
\begin{array}{lll}
\mathcal{S} & ::= & V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S}) \\
V & ::= & \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"}
\end{array}
$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

Bottom-Up Enumeration

$P_1 = \{\texttt{x}, \epsilon, \texttt{"␣Conf"}, \texttt{"␣City"}\}$ $\qquad P_2 = \emptyset$

$$
\begin{aligned}
P_3 = \{&\texttt{x}.\texttt{x}, \texttt{x}.\epsilon, \texttt{x}.\texttt{"␣Conf"}, \texttt{x}.\texttt{"␣City"}, \epsilon.\texttt{x}, \epsilon.\epsilon, \epsilon.\texttt{"␣Conf"}, \epsilon.\texttt{"␣City"}, \\
&\texttt{"␣Conf"}.\texttt{x}, \texttt{"␣Conf"}.\epsilon, \texttt{"␣Conf"}.\texttt{"␣Conf"}, \texttt{"␣City"}.\texttt{"␣City"}, \\
&\texttt{"␣Conf"}.\texttt{"␣City"}, \texttt{"␣City"}.\texttt{x}, \texttt{"␣City"}.\epsilon, \texttt{"␣City"}.\texttt{"␣Conf"}\}
\end{aligned}
$$

$P_4 = \{\texttt{r}(\texttt{x}, \texttt{x}, \texttt{"␣City"}), \texttt{r}(\texttt{x}, \texttt{"␣City"}, \texttt{"␣Conf"}), \texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon), \ldots\}$

$P_5 = \{\ldots\}$ $\qquad P_6 = \{\ldots\}$

Bottom-Up Enumeration

$$\begin{array}{lcl} \mathcal{S} & ::= & V \mid \texttt{replace}(\mathcal{S}, \mathcal{S}, \mathcal{S}) \mid \texttt{concat}(\mathcal{S}, \mathcal{S}) \\ V & ::= & \texttt{x} \mid \epsilon \mid \texttt{"\textvisiblespace Conf"} \mid \texttt{"\textvisiblespace City"} \end{array}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

$$P_1 = \{\texttt{x}, \epsilon, \texttt{"\textvisiblespace Conf"}, \texttt{"\textvisiblespace City"}\} \qquad P_2 = \emptyset$$

$$\begin{aligned} P_3 = \{&\texttt{x.x}, \texttt{x.}\epsilon, \texttt{x."\textvisiblespace Conf"}, \texttt{x."\textvisiblespace City"}, \epsilon\texttt{.x}, \epsilon.\epsilon, \epsilon\texttt{."\textvisiblespace Conf"}, \epsilon\texttt{."\textvisiblespace City"}, \\ &\texttt{"\textvisiblespace Conf".x}, \texttt{"\textvisiblespace Conf".}\epsilon, \texttt{"\textvisiblespace Conf"."\textvisiblespace Conf"}, \texttt{"\textvisiblespace City"."\textvisiblespace City"}, \\ &\texttt{"\textvisiblespace Conf"."\textvisiblespace City"}, \texttt{"\textvisiblespace City".x}, \texttt{"\textvisiblespace City".}\epsilon, \texttt{"\textvisiblespace City"."\textvisiblespace Conf"}\} \end{aligned}$$

$$P_4 = \{\texttt{r}(\texttt{x}, \texttt{x}, \texttt{"\textvisiblespace City"}), \texttt{r}(\texttt{x}, \texttt{"\textvisiblespace City"}, \texttt{"\textvisiblespace Conf"}), \texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon), \ldots\}$$

$$P_5 = \{\ldots\} \qquad P_6 = \{\ldots\} \qquad P_7 = \{\ldots, \texttt{r}(\texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon), \texttt{"\textvisiblespace City"}, \epsilon), \ldots\}$$

$$\begin{array}{lcl} \mathcal{S} & ::= & V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S}) \\ V & ::= & \texttt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"} \end{array}$$

Bottom-Up Enumeration

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

$P_1 = \{\texttt{x}, \epsilon, \texttt{"␣Conf"}, \texttt{"␣City"}\}$ $\qquad$ $P_2 = \emptyset$

$$\begin{aligned} P_3 = \{&\texttt{x.x}, \texttt{x}.\epsilon, \texttt{x."␣Conf"}, \texttt{x."␣City"}, \epsilon.\texttt{x}, \epsilon.\epsilon, \epsilon.\texttt{"␣Conf"}, \epsilon.\texttt{"␣City"}, \\ &\texttt{"␣Conf".x}, \texttt{"␣Conf"}.\epsilon, \texttt{"␣Conf"."␣Conf"}, \texttt{"␣City"."␣City"}, \\ &\texttt{"␣Conf"."␣City"}, \texttt{"␣City".x}, \texttt{"␣City"}.\epsilon, \texttt{"␣City"."␣Conf"}\} \end{aligned}$$

$$P_4 = \{\texttt{r}(\texttt{x}, \texttt{x}, \texttt{"␣City"}), \texttt{r}(\texttt{x}, \texttt{"␣City"}, \texttt{"␣Conf"}), \texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon), \ldots\}$$

$$P_5 = \{\ldots\} \qquad P_6 = \{\ldots\} \qquad P_7 = \{\ldots, \texttt{r}(\texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon), \texttt{"␣City"}, \epsilon), \ldots\}$$

$$\mathcal{S} \;::=\; V \mid \texttt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \texttt{concat}(\mathcal{S},\mathcal{S})$$

$$V \;::=\; \texttt{x} \mid \epsilon \mid \texttt{"\textvisiblespace Conf"} \mid \texttt{"\textvisiblespace City"}$$

Bottom-Up Enumeration

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

$$P_1 = \{\texttt{x}, \epsilon, \texttt{"\textvisiblespace Conf"}, \texttt{"\textvisiblespace City"}\} \qquad P_2 = \emptyset$$

$$\begin{aligned}P_3 = \{&\texttt{x}.\texttt{x}, \texttt{x}.\epsilon, \texttt{x}.\texttt{"\textvisiblespace Conf"}, \texttt{x}.\texttt{"\textvisiblespace City"}, \epsilon.\texttt{x}, \epsilon.\epsilon, \epsilon.\texttt{"\textvisiblespace Conf"}, \epsilon.\texttt{"\textvisiblespace City"},\\ &\texttt{"\textvisiblespace Conf"}.\texttt{x}, \texttt{"\textvisiblespace Conf"}.\epsilon, \texttt{"\textvisiblespace Conf"}.\texttt{"\textvisiblespace Conf"}, \texttt{"\textvisiblespace City"}.\texttt{"\textvisiblespace City"},\\ &\texttt{"\textvisiblespace Conf"}.\texttt{"\textvisiblespace City"}, \texttt{"\textvisiblespace City"}.\texttt{x}, \texttt{"\textvisiblespace City"}.\epsilon, \texttt{"\textvisiblespace City"}.\texttt{"\textvisiblespace Conf"}\}\end{aligned}$$

$$P_4 = \{\texttt{r}(\texttt{x}, \texttt{x}, \texttt{"\textvisiblespace City"}), \texttt{r}(\texttt{x}, \texttt{"\textvisiblespace City"}, \texttt{"\textvisiblespace Conf"}), \texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon), \ldots\}$$

$$P_5 = \{\ldots\} \qquad P_6 = \{\ldots\} \qquad P_7 = \{\ldots, \texttt{r}(\texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon), \texttt{"\textvisiblespace City"}, \epsilon), \ldots\}$$

Exponential Blowup!

$$\mathcal{S} \quad ::= \quad V \mid \mathtt{replace}(\mathcal{S},\mathcal{S},\mathcal{S}) \mid \mathtt{concat}(\mathcal{S},\mathcal{S})$$

$$V \quad ::= \quad \mathtt{x} \mid \epsilon \mid \texttt{"␣Conf"} \mid \texttt{"␣City"}$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

Bottom-Up Enumeration

$$P_1 = \{\mathtt{x}, \epsilon, \texttt{"␣Conf"}, \texttt{"␣City"}\} \qquad P_2 = \emptyset$$

$$\begin{aligned} P_3 = \{&\mathtt{x}.\mathtt{x}, \mathtt{x}.\epsilon, \mathtt{x}.\texttt{"␣Conf"}, \mathtt{x}.\texttt{"␣City"}, \epsilon.\mathtt{x}, \epsilon.\epsilon, \epsilon.\texttt{"␣Conf"}, \epsilon.\texttt{"␣City"},\\ &\texttt{"␣Conf"}.\mathtt{x}, \texttt{"␣Conf"}.\epsilon, \texttt{"␣Conf"}.\texttt{"␣Conf"}, \texttt{"␣City"}.\texttt{"␣City"},\\ &\texttt{"␣Conf"}.\texttt{"␣City"}, \texttt{"␣City"}.\mathtt{x}, \texttt{"␣City"}.\epsilon, \texttt{"␣City"}.\texttt{"␣Conf"}\}\end{aligned}$$

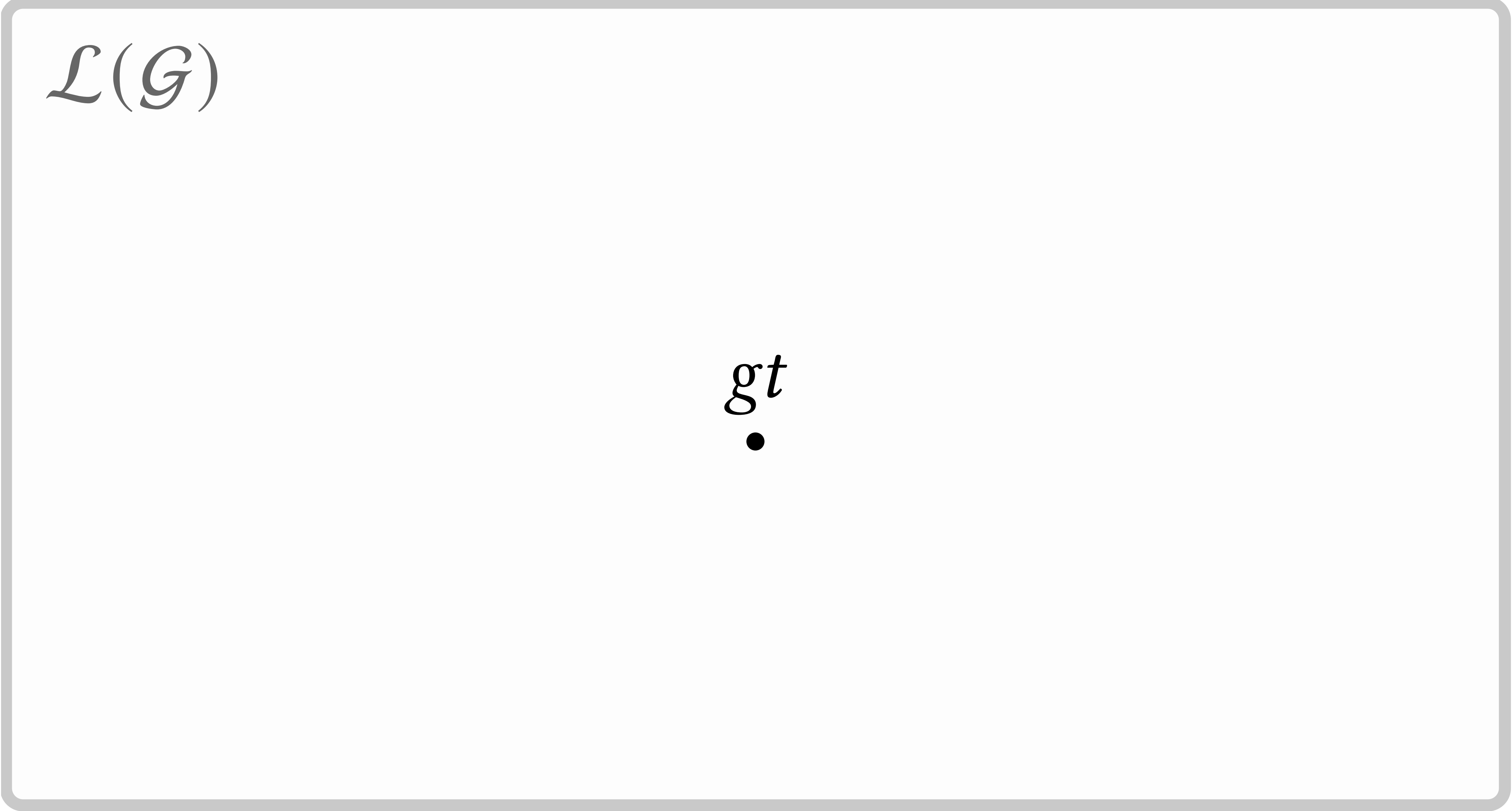
$$P_4 = \{\mathtt{r}(\mathtt{x}, \mathtt{x}, \texttt{"␣City"}), \mathtt{r}(\mathtt{x}, \texttt{"␣City"}, \texttt{"␣Conf"}), \mathtt{r}(\mathtt{x}, \texttt{"␣Conf"}, \epsilon), \ldots\}$$

$$P_5 = \{\ldots\} \qquad P_6 = \{\ldots\} \qquad P_7 = \{\ldots, \mathtt{r}(\mathtt{r}(\mathtt{x}, \texttt{"␣Conf"}, \epsilon), \texttt{"␣City"}, \epsilon), \ldots\}$$

Exponential Blowup!

Method	P_1	P_2	P_3	P_4	P_5	P_6	P_7
No Pruning or Factorization	4	-	16	64	128	1280	4352

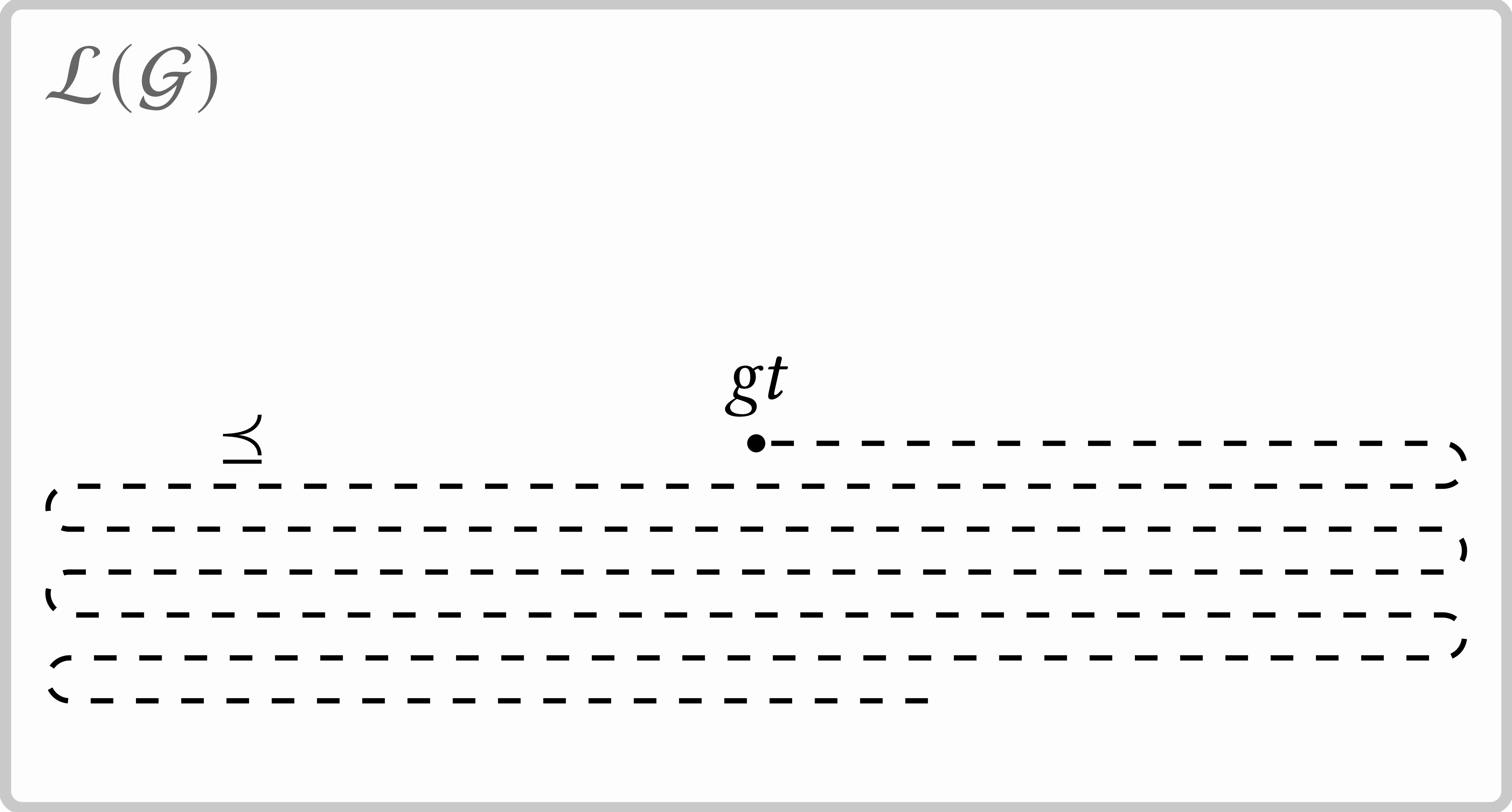
Enumeration, Factorization, Pruning



Enumeration, Factorization, Pruning

Enumeration Order:

Size-based

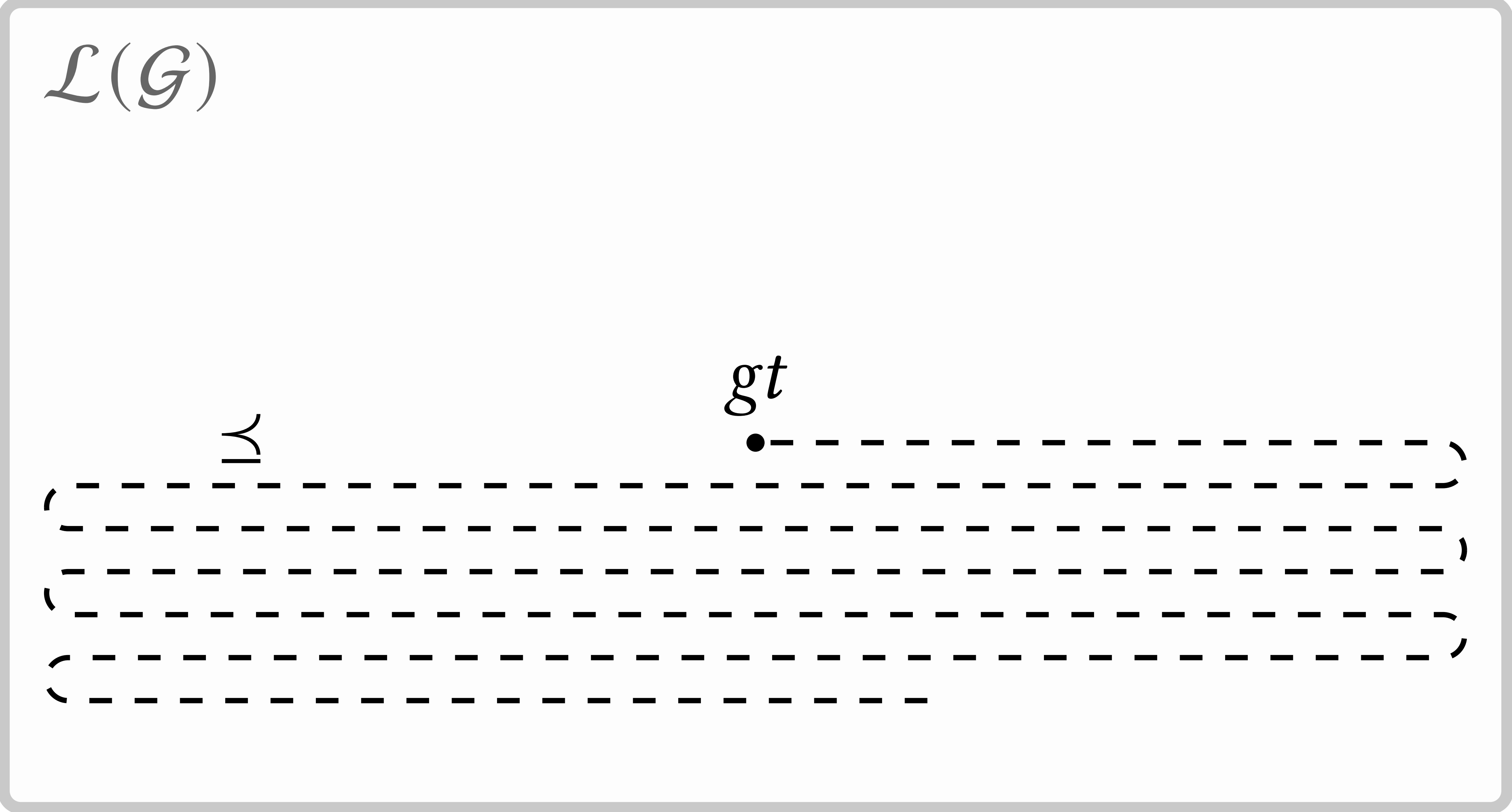


Enumeration, Factorization, Pruning

Enumeration Order:

Size-based

Deduction [Alur et al. 2017, Lee 2021, Yoon et al. 2023, Ding and Qiu 2024, Ding and Qiu 2025]



Deduction

Deduction

ϵ

Deduction

$\epsilon \preceq$ `"␣City"`

Deduction

$\epsilon \preceq$ `"␣City"` $\preceq \ldots$

$\ldots$

Deduction

$$\epsilon \preceq \texttt{"␣City"} \preceq \ldots$$

$$\ldots \preceq \texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon)$$

Deduction

$$\epsilon \preceq \texttt{"␣City"} \preceq \ldots$$

$$\ldots \preceq \texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon) \preceq \ldots$$

Deduction

$$\epsilon \preceq \texttt{"\textvisiblespace City"} \preceq \dots$$

$$\dots \preceq \texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon) \preceq \dots \preceq \texttt{r}(\texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon), \texttt{"\textvisiblespace City"}, \epsilon)$$

Deduction

$$\epsilon \preceq \texttt{"␣City"} \preceq \ldots$$

$$\ldots \preceq \underline{\texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon)} \preceq \ldots \preceq \texttt{r}(\texttt{r}(\texttt{x}, \texttt{"␣Conf"}, \epsilon), \texttt{"␣City"}, \epsilon)$$

Deduction

$$\epsilon \preceq \texttt{"\textvisiblespace City"} \preceq \ldots$$

$$\ldots \preceq \underline{\texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon)} \preceq \ldots \preceq \texttt{r}(\texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon), \texttt{"\textvisiblespace City"}, \epsilon)$$

Deduction

$$\epsilon \preceq \texttt{"\textvisiblespace City"} \preceq \ldots$$

$$\ldots \preceq \texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon) \qquad \texttt{r}(\texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon), \texttt{"\textvisiblespace City"}, \epsilon)$$

Deduction

$$\epsilon \preceq \texttt{"\textvisiblespace City"} \preceq \ldots$$

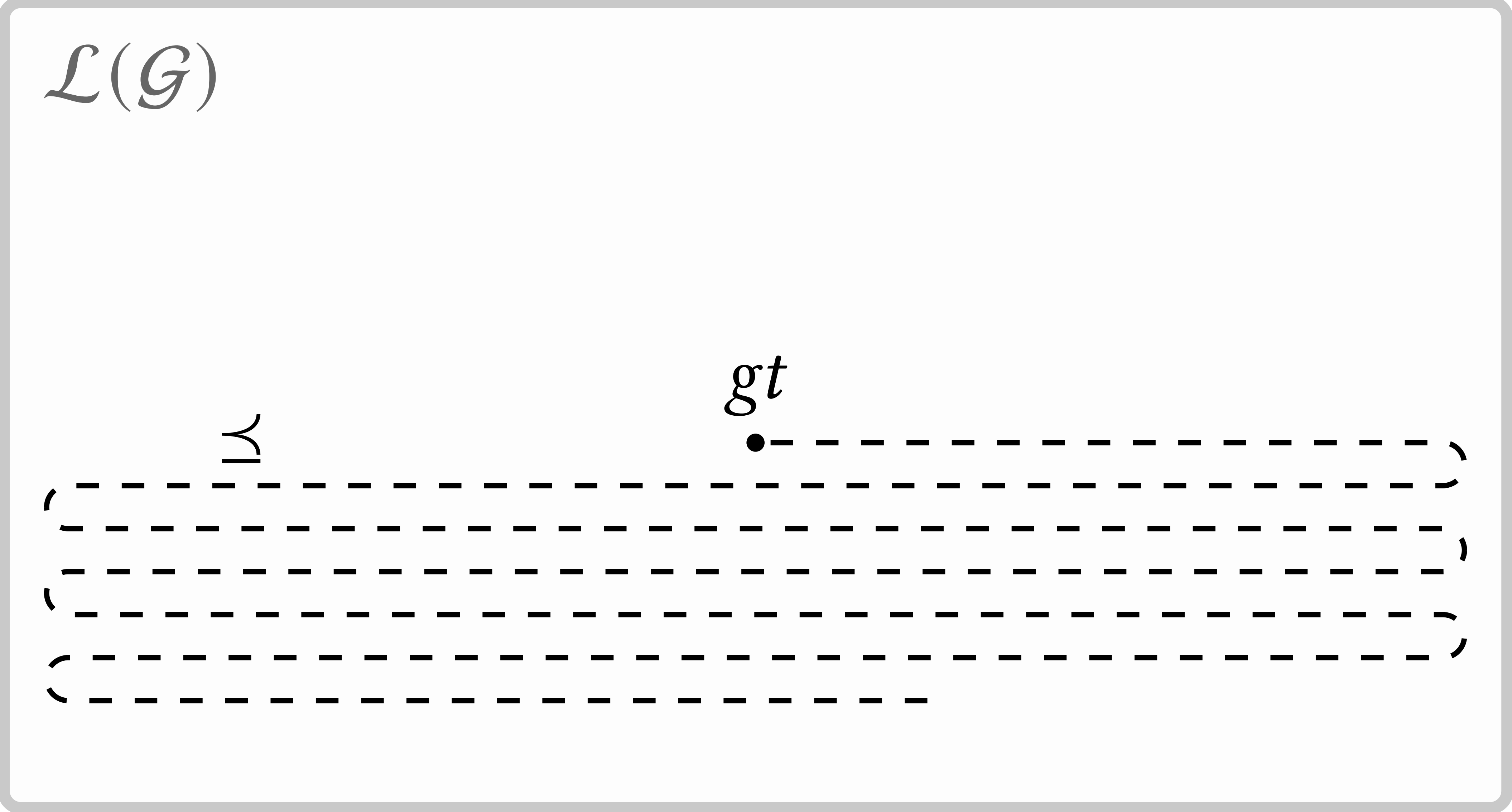
$$\ldots \preceq \underline{\texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon)} \preceq \texttt{r}(\texttt{r}(\texttt{x}, \texttt{"\textvisiblespace Conf"}, \epsilon), \texttt{"\textvisiblespace City"}, \epsilon)$$

Enumeration, Factorization, Pruning

Enumeration Order:

Size-based

Deduction [Alur et al. 2017, Lee 2021, Yoon et al. 2023, Ding and Qiu 2024, Ding and Qiu 2025]



Understanding 2: Existing approaches

have a way to **factorize** the search space

Enumeration, Factorization, Pruning

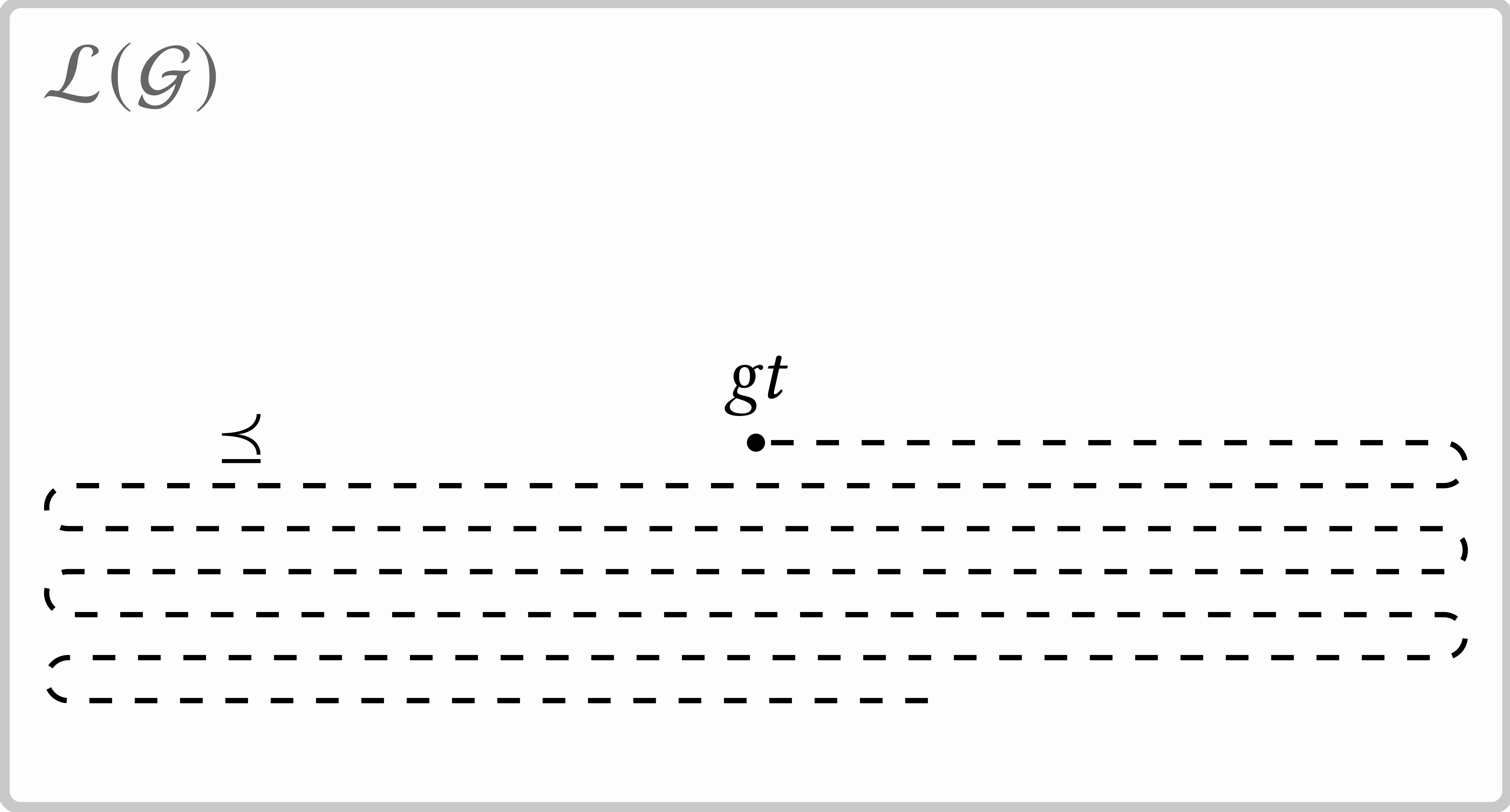
Enumeration Order:

Size-based

Deduction [Alur et al. 2017, Lee 2021, Yoon et al. 2023, Ding and Qiu 2024, Ding and Qiu 2025]

Factorizations:

Observational Equivalence Factorization [Udapa et al. 2013, Albarghouthi et al. 2013]

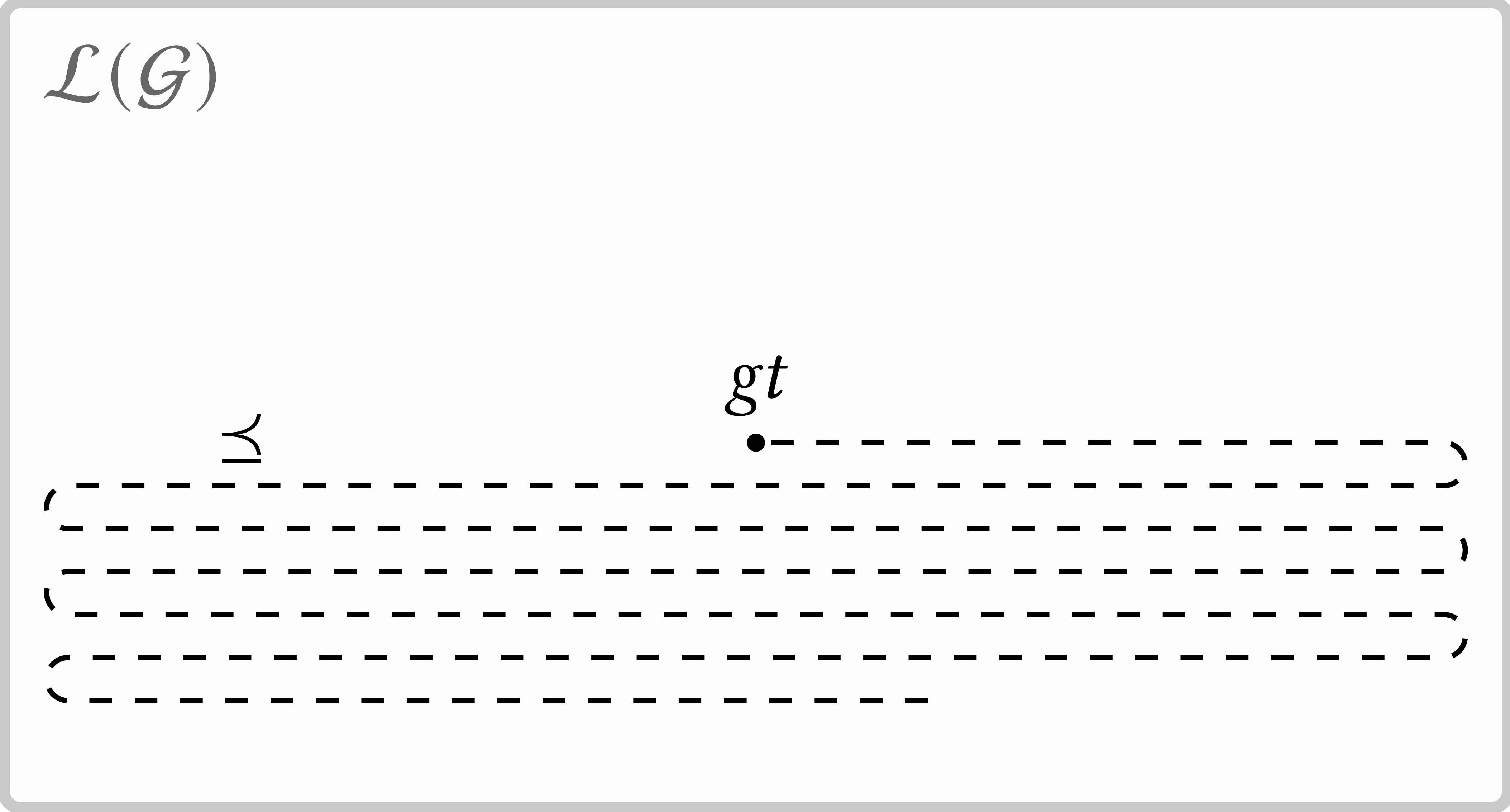


Observational Equivalence Factorization

Observational Equivalence:

$$\forall i \in In : \mathsf{prog}_1(i) = \mathsf{prog}_2(i)$$

$$\mathsf{x} \approx \mathsf{x}\,.\,\epsilon$$



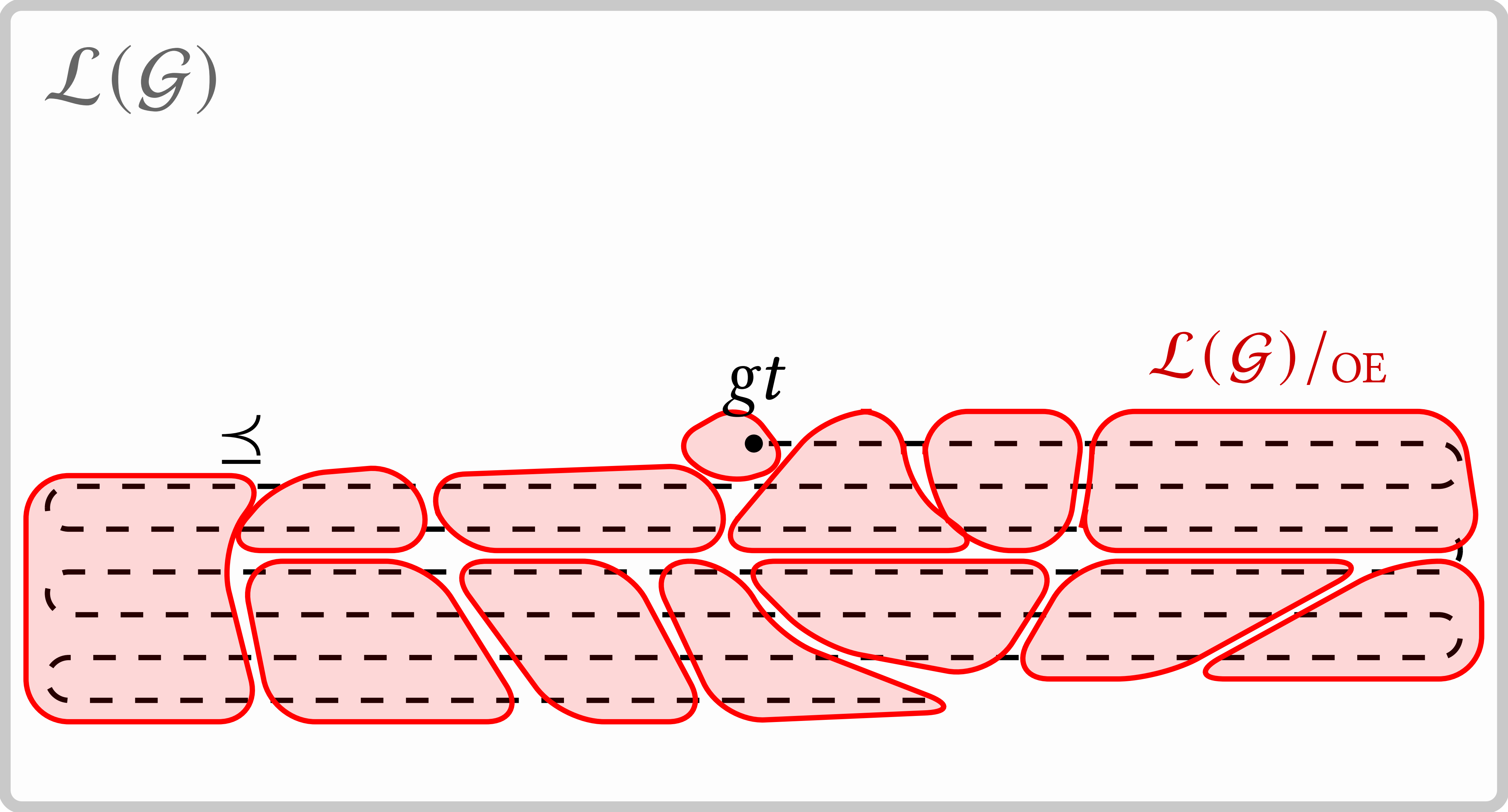
Observational Equivalence Factorization

Observational Equivalence:

$$\forall i \in In : \mathsf{prog}_1(i) = \mathsf{prog}_2(i)$$

$$\mathsf{x} \approx \mathsf{x}\,.\,\epsilon$$

Factorizes search space


$\mathcal{L}(\mathcal{G})$
gt
$\preceq$
$\mathcal{L}(\mathcal{G})/_{\mathrm{OE}}$

Observational Equivalence Factorization

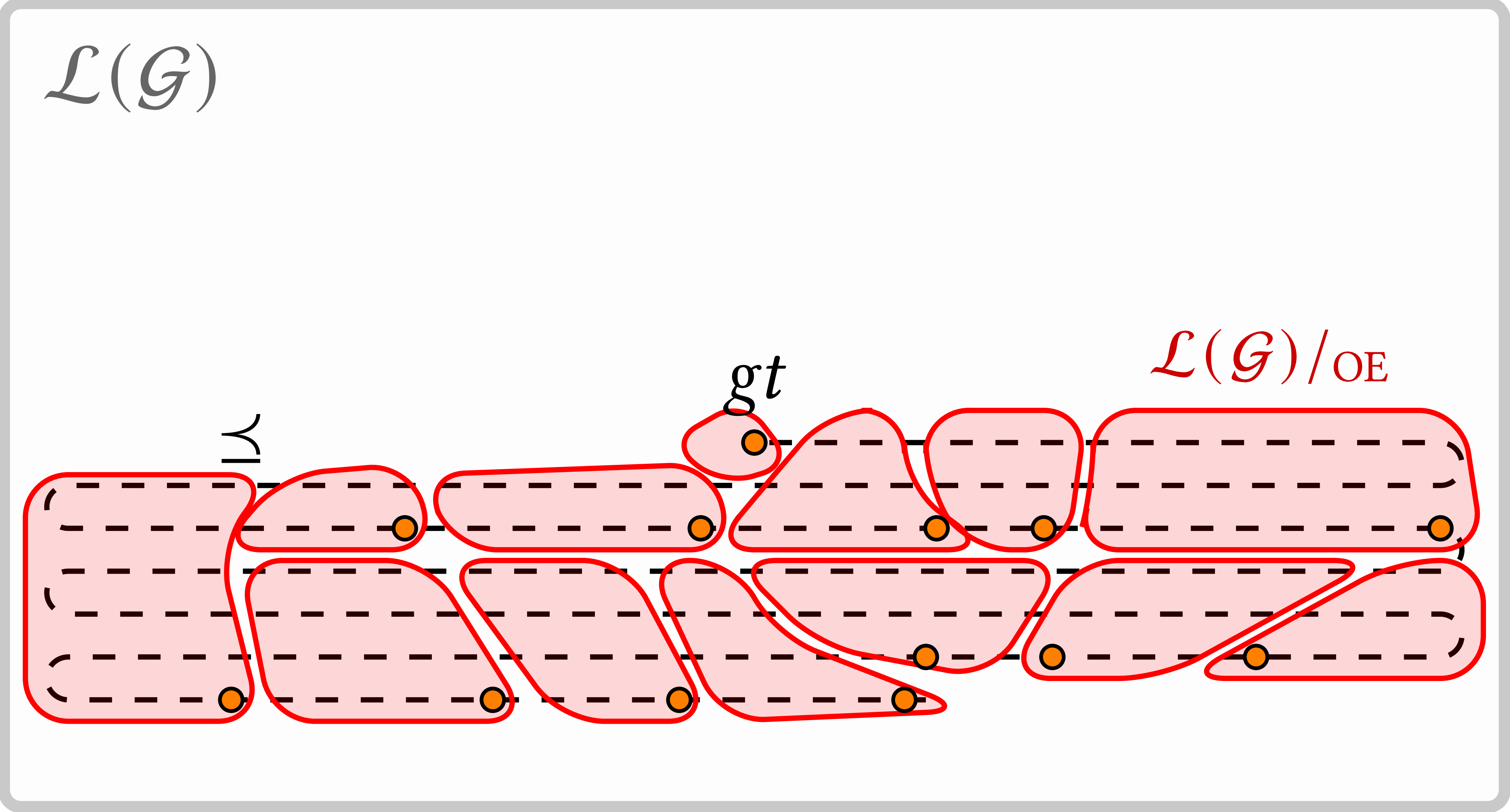
Observational Equivalence:

$$\forall i \in In : \mathsf{prog}_1(i) = \mathsf{prog}_2(i)$$

$$\mathsf{x} \approx \mathsf{x} \,.\, \epsilon$$

Factorizes search space

Only keep one representative per class


$\mathcal{L}(\mathcal{G})$
$\preceq$
gt
$\mathcal{L}(\mathcal{G})/_{\mathrm{OE}}$

Observational Equivalence Factorization

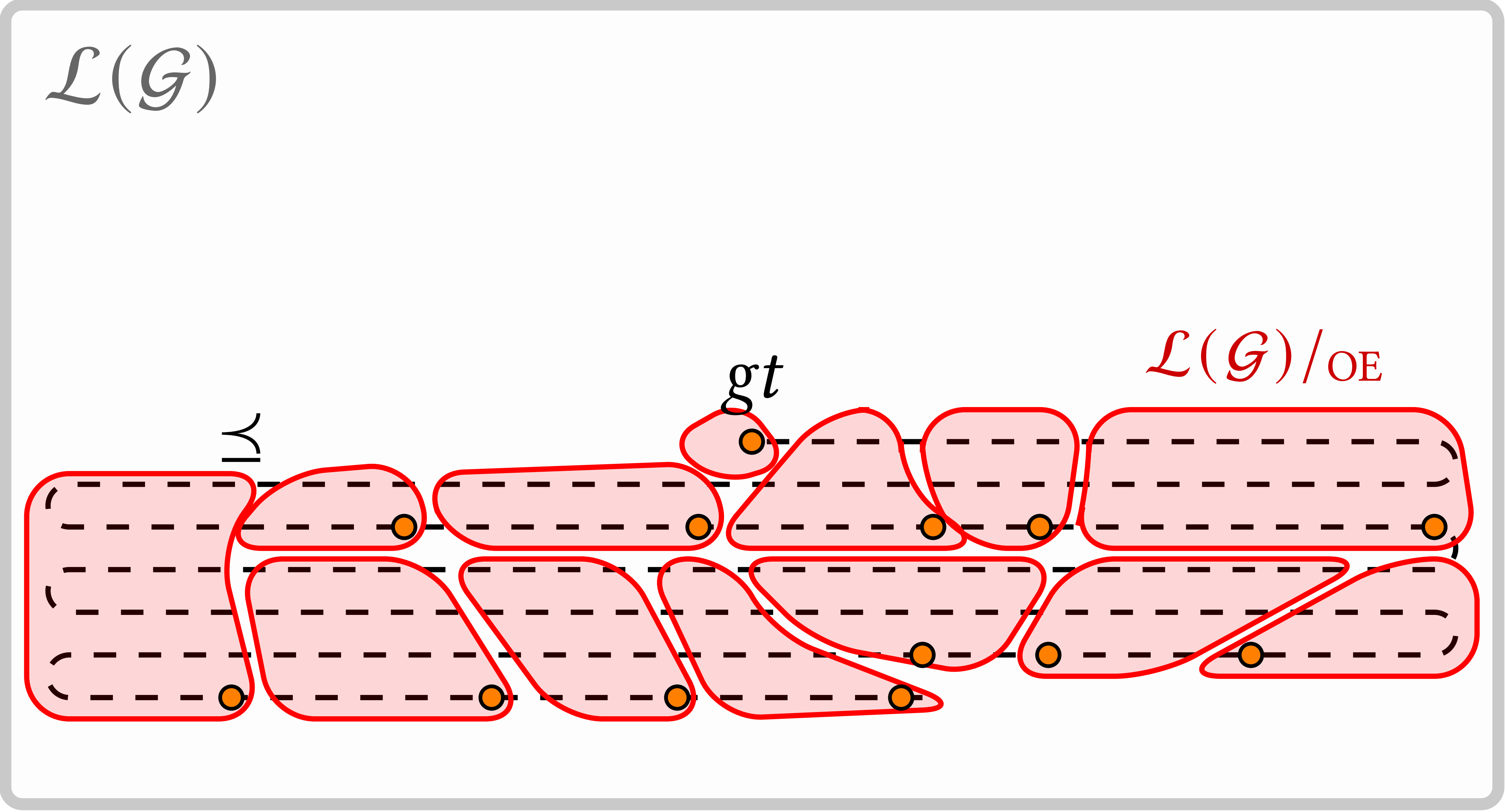
Observational Equivalence:

$$\forall i \in In : \text{prog}_1(i) = \text{prog}_2(i)$$

$$\cancel{\text{x} \approx \text{x} \pm \epsilon}$$

Factorizes search space

Only keep one representative per class


$\mathcal{L}(\mathcal{G})$
gt
$\preceq$
$\mathcal{L}(\mathcal{G})/_{\text{OE}}$

Observational Equivalence Factorization

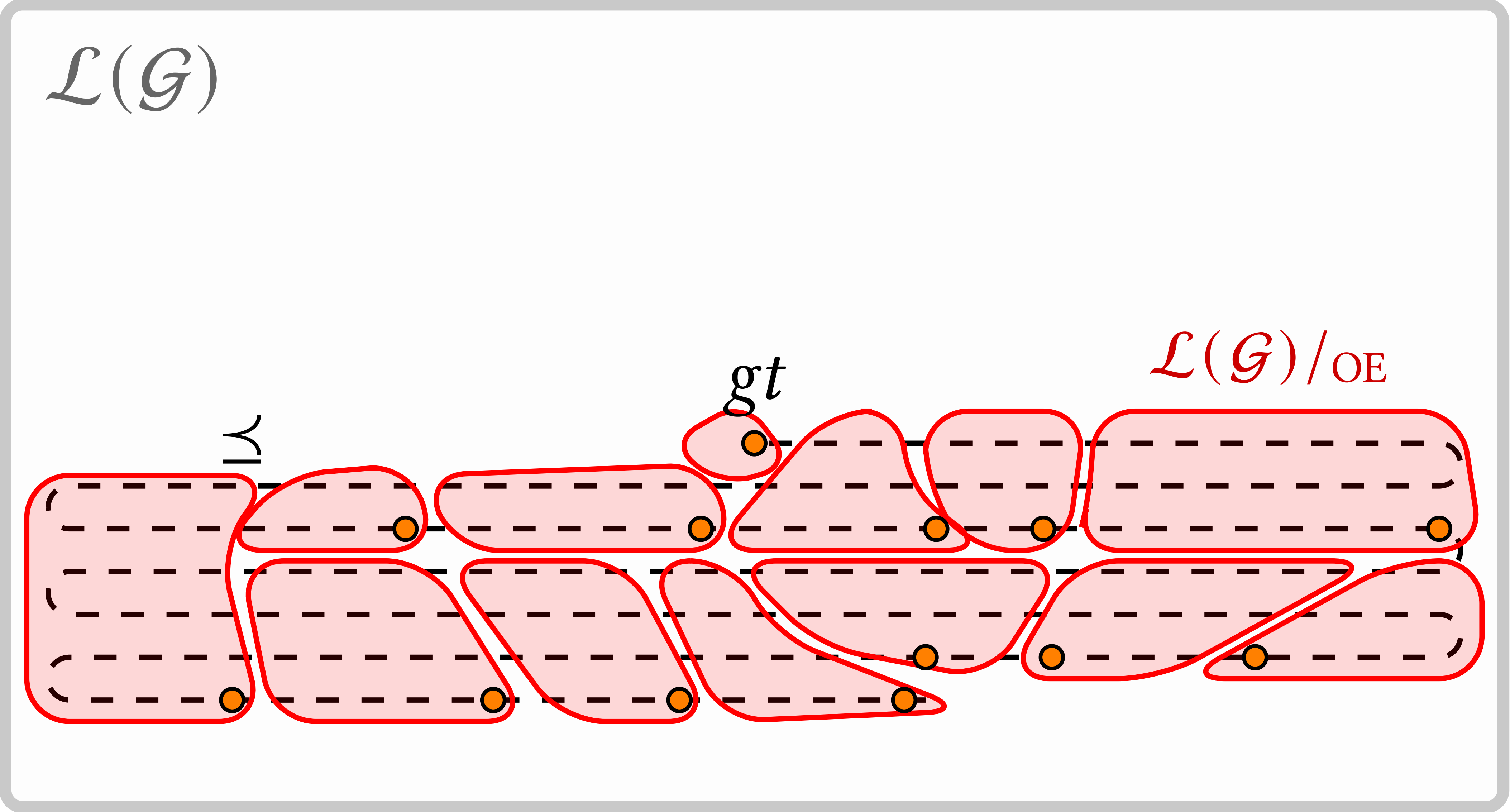
Observational Equivalence:

$$\forall i \in In : \mathsf{prog}_1(i) = \mathsf{prog}_2(i)$$

$\mathsf{x} \approx \mathsf{x}$ ✗ ϵ

Factorizes search space

Only keep one representative per class


$\mathcal{L}(\mathcal{G})$
gt
$\mathcal{L}(\mathcal{G})/_{\mathrm{OE}}$
$\preceq$

Method	P_1	P_2	P_3	P_4	P_5	P_6	P_7
No Pruning or Factorization	4	-	16	64	128	1280	4352
OE Factorization	4	-	9	6	27	56	119

Enumeration, Factorization, Pruning

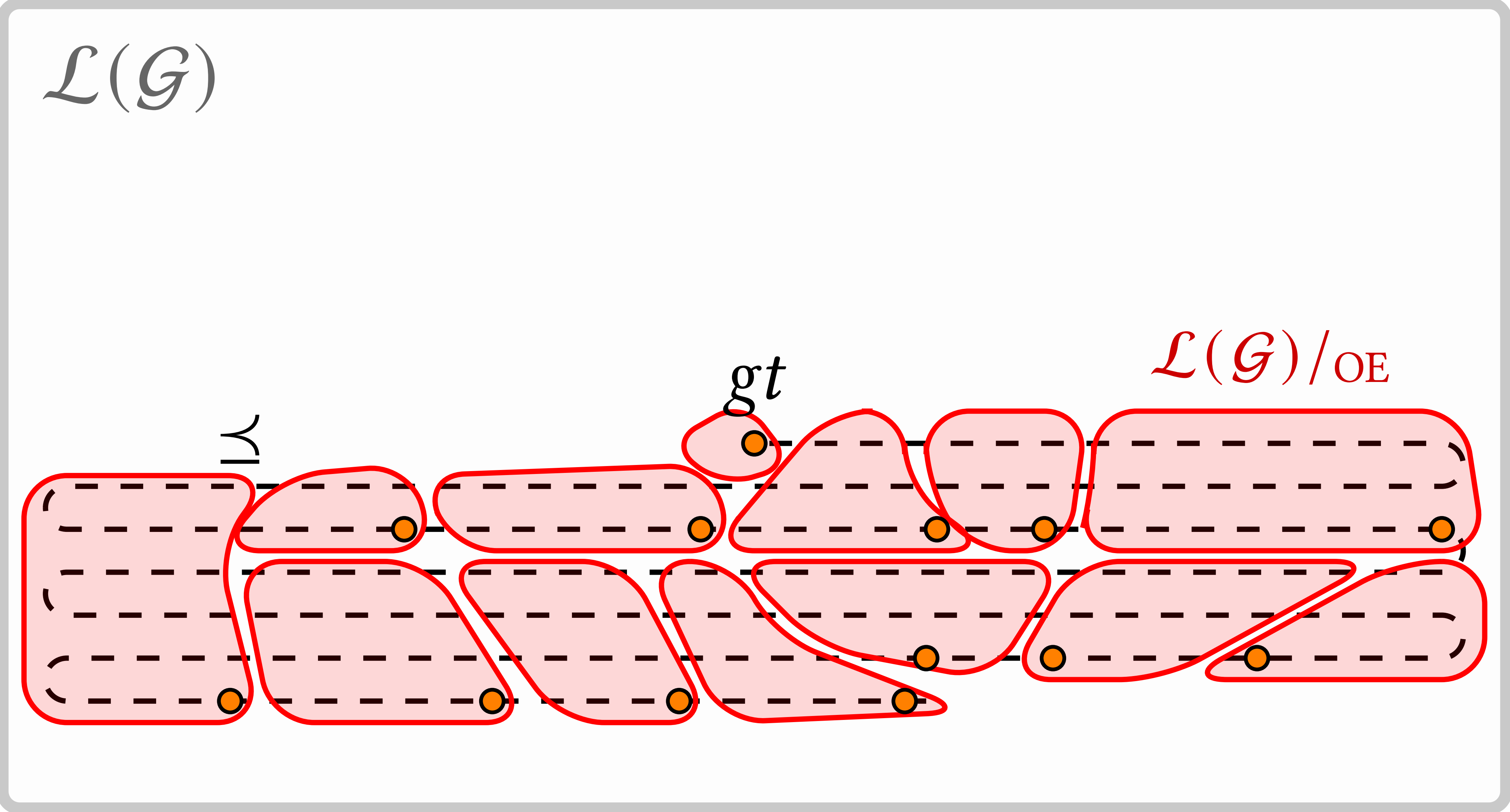
Enumeration Order:

Size-based

Deduction [Alur et al. 2017, Lee 2021, Yoon et al. 2023, Ding and Qiu 2024, Ding and Qiu 2025]

Factorizations:

Observational Equivalence Factorization [Udapa et al. 2013, Albarghouthi et al. 2013]


$\mathcal{L}(\mathcal{G})$
$\preceq$
gt
$\mathcal{L}(\mathcal{G})/_{\text{OE}}$

Enumeration, Factorization, Pruning

Enumeration Order:

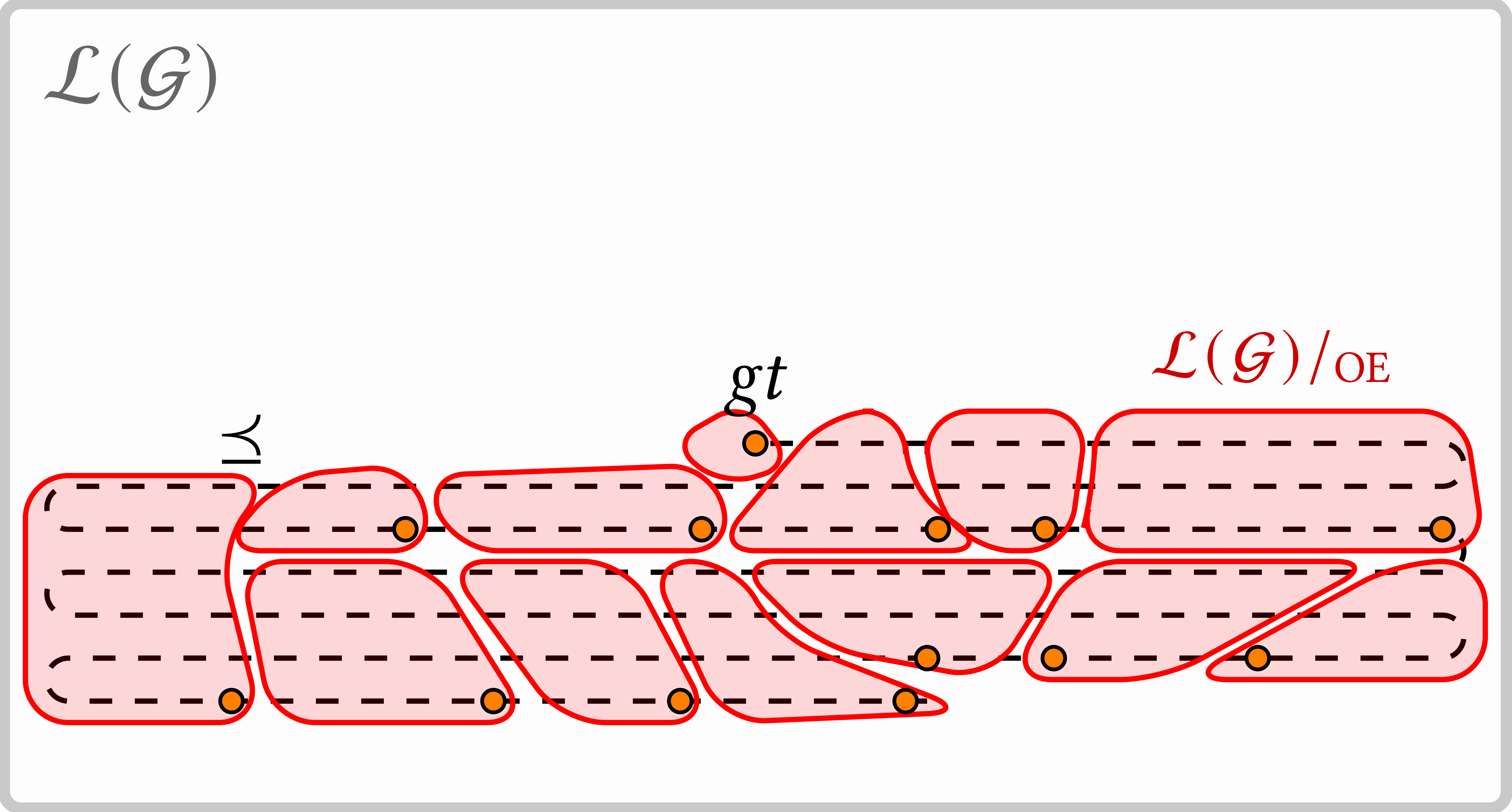
Size-based

Deduction [Alur et al. 2017, Lee 2021, Yoon et al. 2023, Ding and Qiu 2024, Ding and Qiu 2025]

Factorizations:

Observational Equivalence Factorization [Udapa et al. 2013, Albarghouthi et al. 2013]

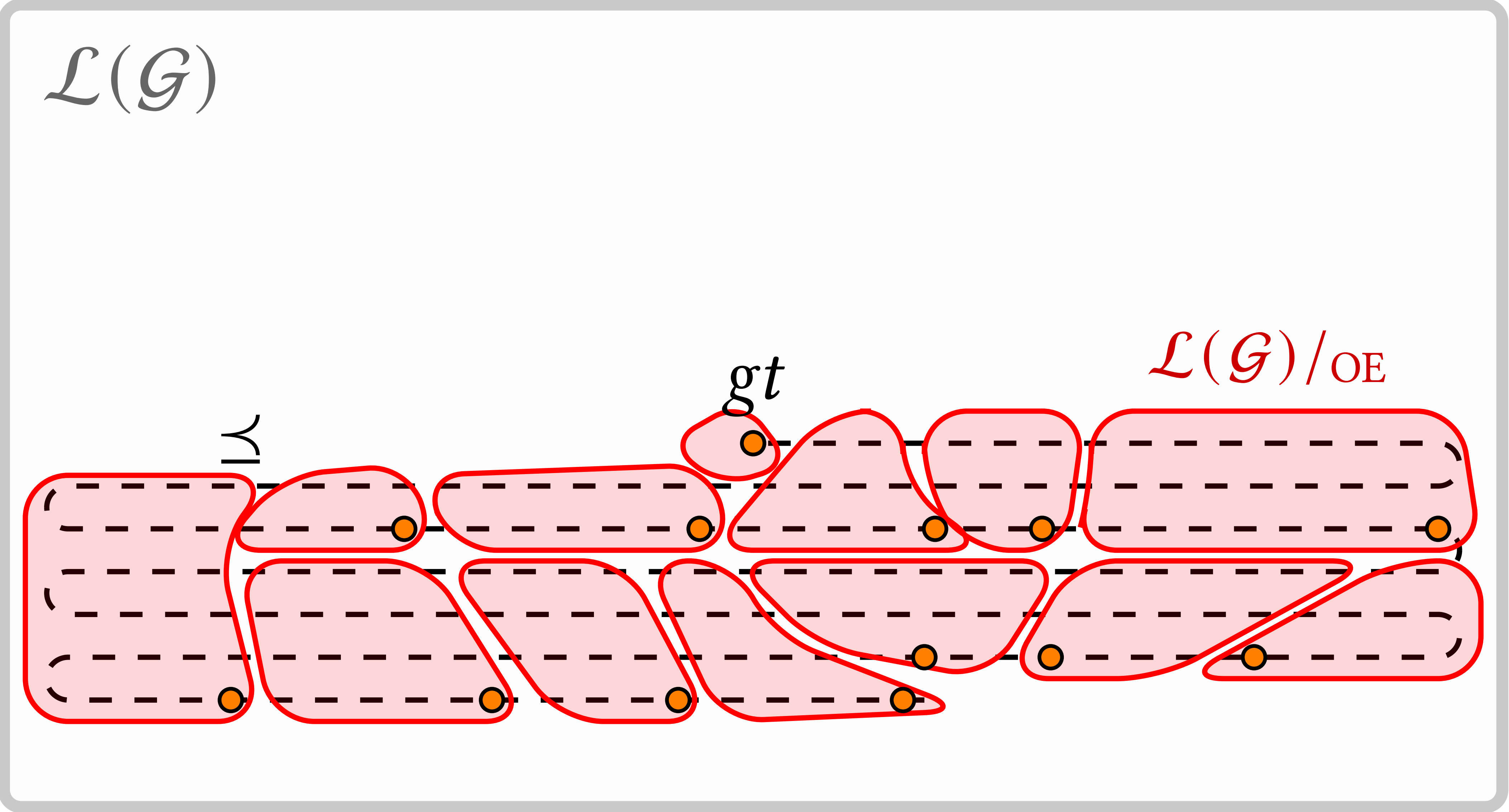
Abstraction [Wang et al. 2018]


$\mathcal{L}(\mathcal{G})$
$\preceq$
gt
$\mathcal{L}(\mathcal{G})/_{\mathrm{OE}}$

Abstraction

Perform OE on abstracted values

$$\forall i \in In : \alpha(\mathsf{prog}_1(i)) = \alpha(\mathsf{prog}_2(i))$$

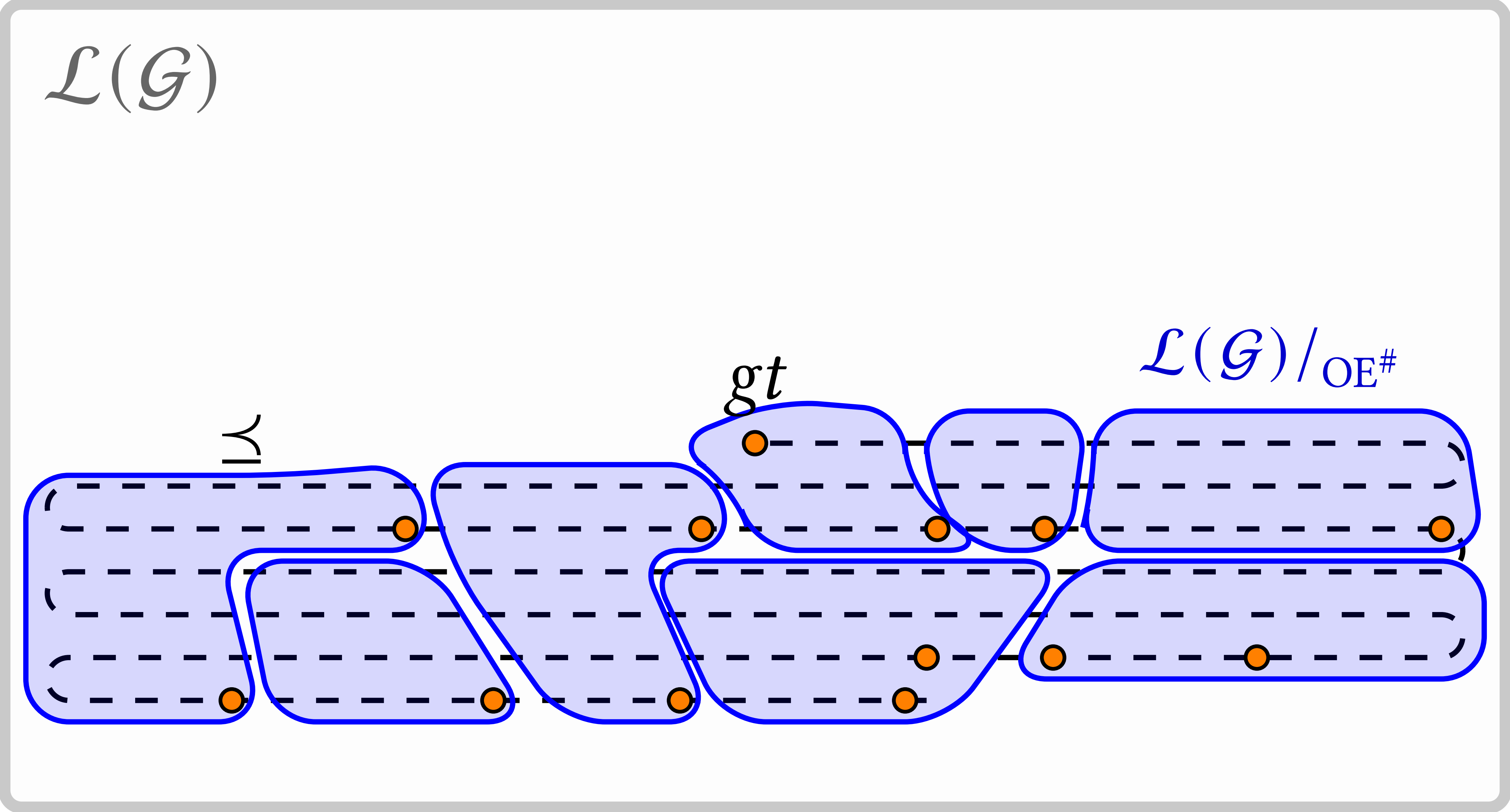

$\mathcal{L}(\mathcal{G})$
gt
$\mathcal{L}(\mathcal{G})/_{\mathrm{OE}}$
$\preceq$

Abstraction

Perform OE on abstracted values

$$\forall i \in In : \alpha(\mathsf{prog}_1(i)) = \alpha(\mathsf{prog}_2(i))$$

Coarser than OE


$\mathcal{L}(\mathcal{G})$
gt
$\mathcal{L}(\mathcal{G})/_{\mathrm{OE}^\#}$
$\preceq$

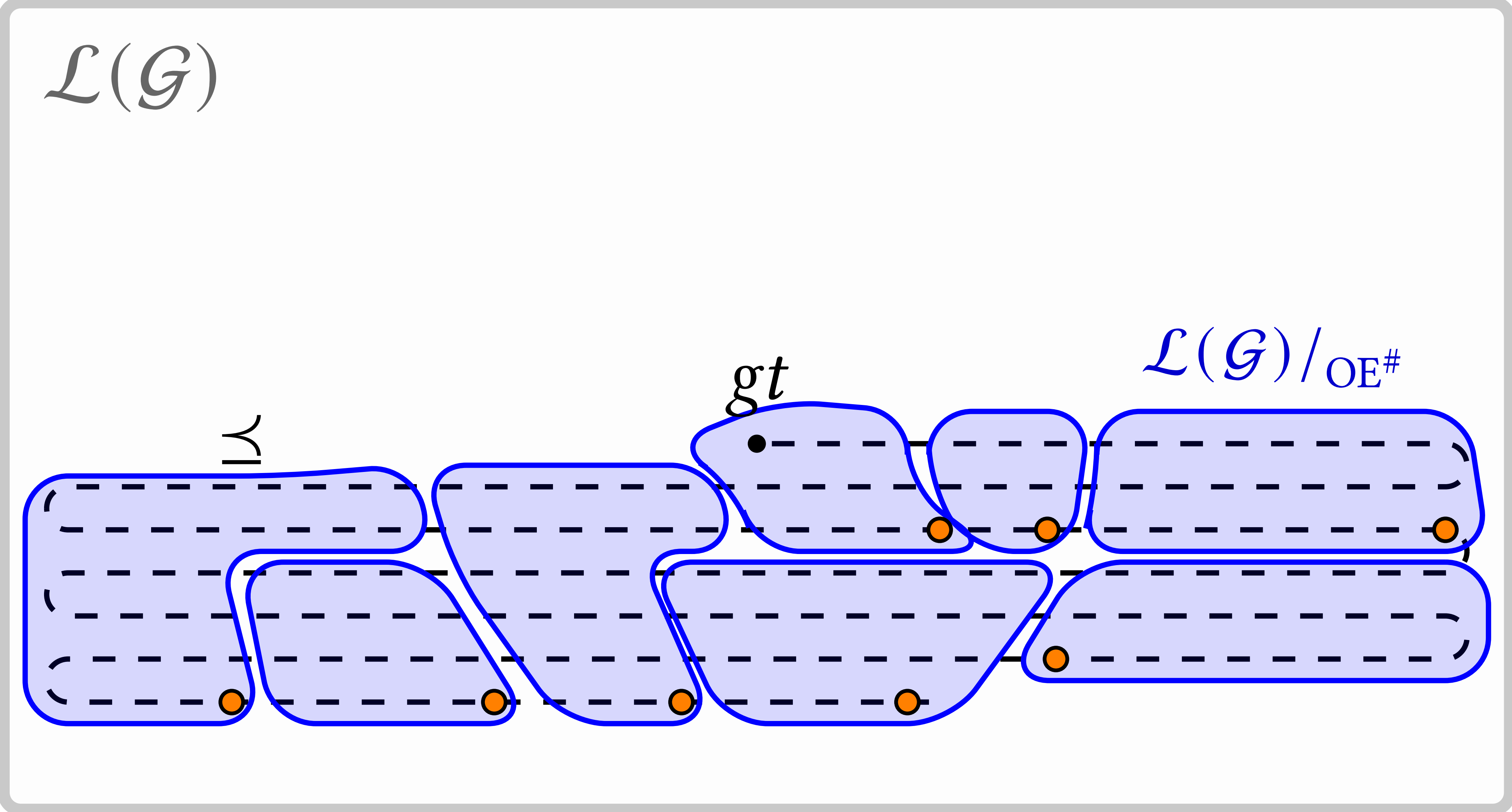
Abstraction

Perform OE on abstracted values

$$\forall i \in In : \alpha(\mathsf{prog}_1(i)) = \alpha(\mathsf{prog}_2(i))$$

Coarser than OE

Only keep one representative per class


$\mathcal{L}(\mathcal{G})$
gt
$\mathcal{L}(\mathcal{G})/_{\mathrm{OE}^\#}$
$\preceq$

Enumeration, Factorization, Pruning

Enumeration Order:

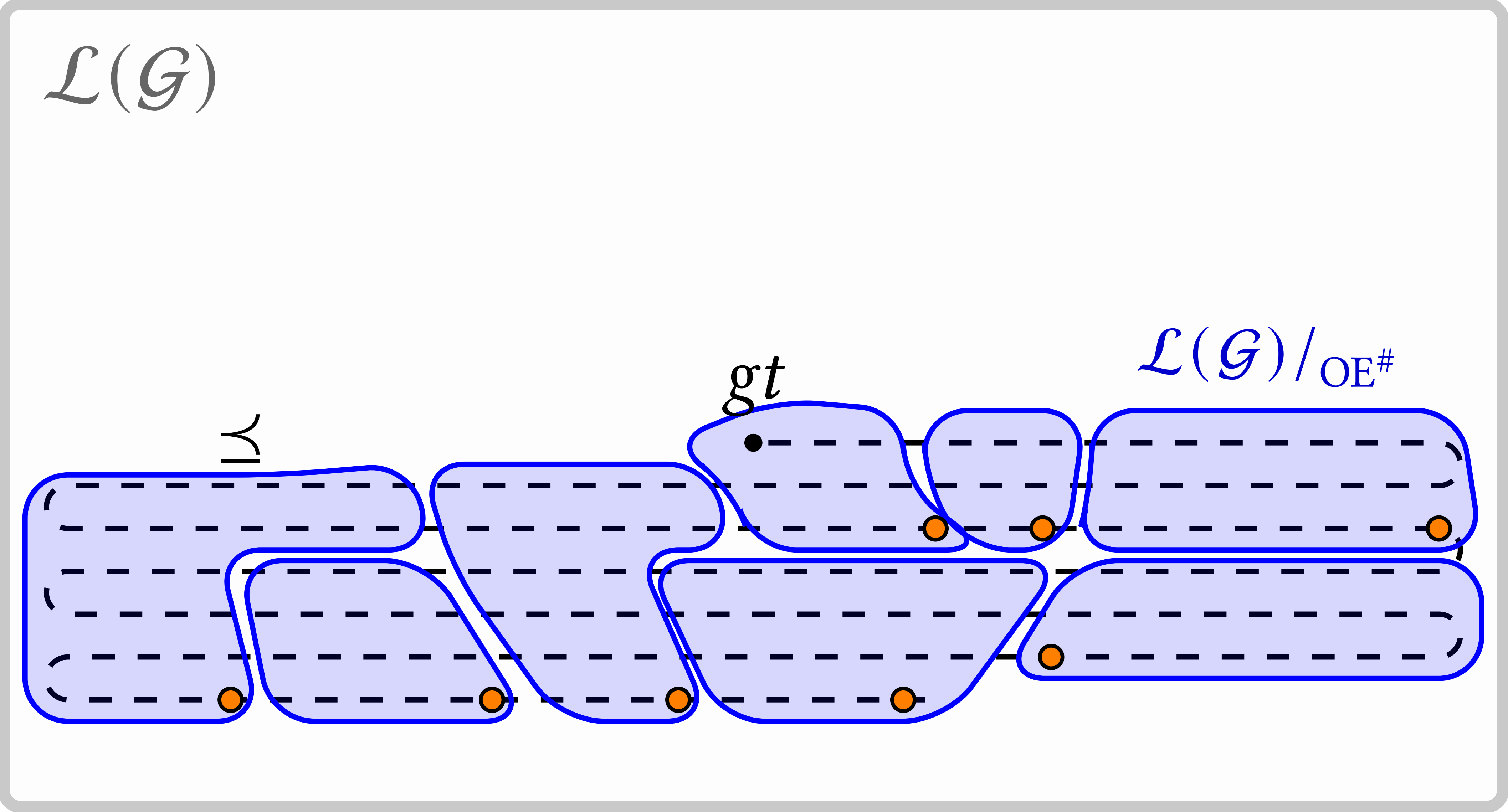
Size-based

Deduction [Alur et al. 2017, Lee 2021, Yoon et al. 2023, Ding and Qiu 2024, Ding and Qiu 2025]

Factorizations:

Observational Equivalence Factorization [Udapa et al. 2013, Albarghouthi et al. 2013]

Abstraction [Wang et al. 2018]


$\mathcal{L}(\mathcal{G})$
$\preceq$
gt
$\mathcal{L}(\mathcal{G})/_{\mathrm{OE}^{\#}}$

Understanding 3: Existing approaches

have a way to **prune** the search space

Enumeration, Factorization, Pruning

Enumeration Order:

Size-based

Deduction [Alur et al. 2017, Lee 2021, Yoon et al. 2023, Ding and Qiu 2024, Ding and Qiu 2025]

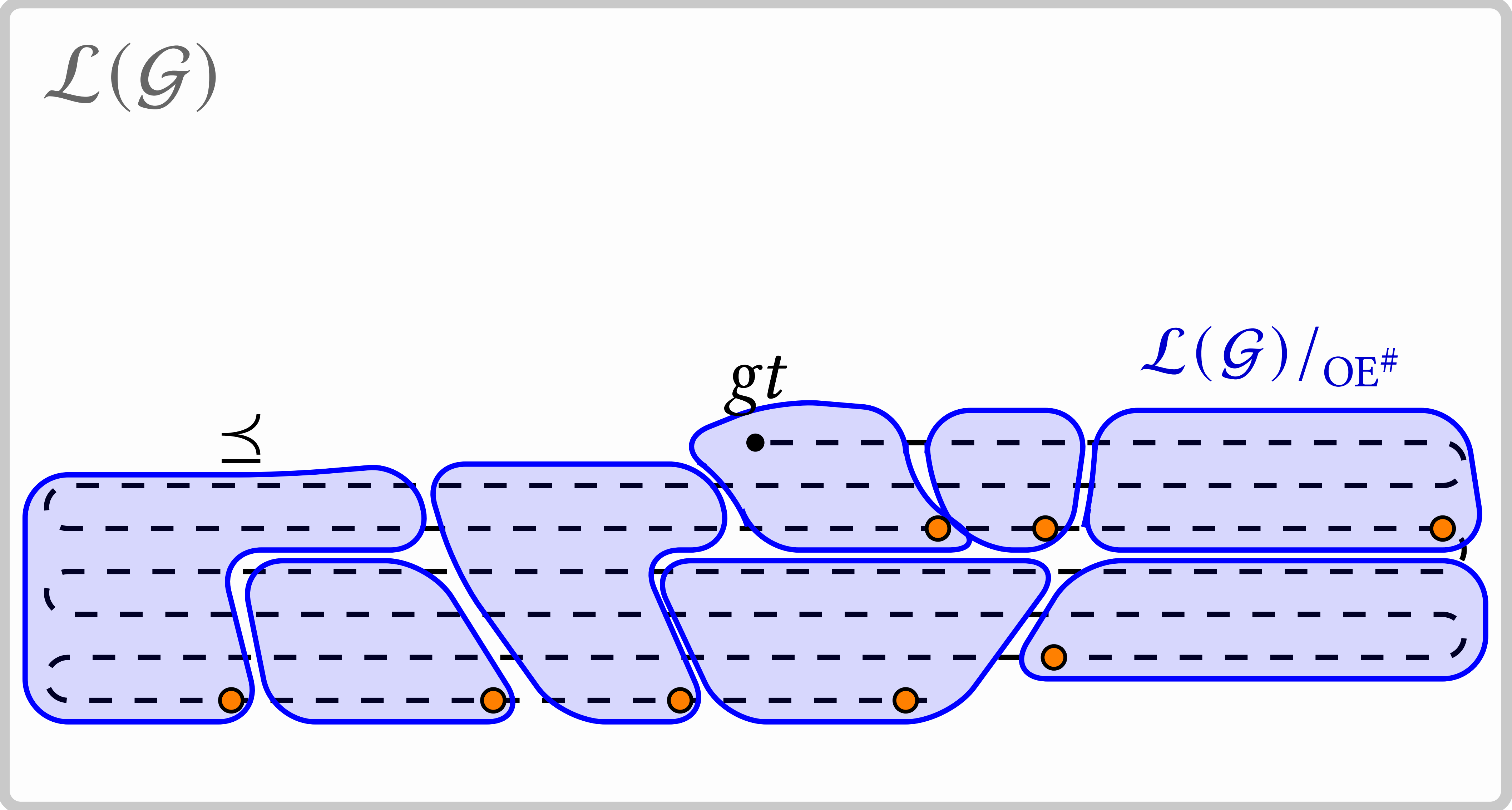
Factorizations:

Observational Equivalence Factorization [Udapa et al. 2013, Albarghouthi et al. 2013]

Abstraction [Wang et al. 2018]

Pruning:

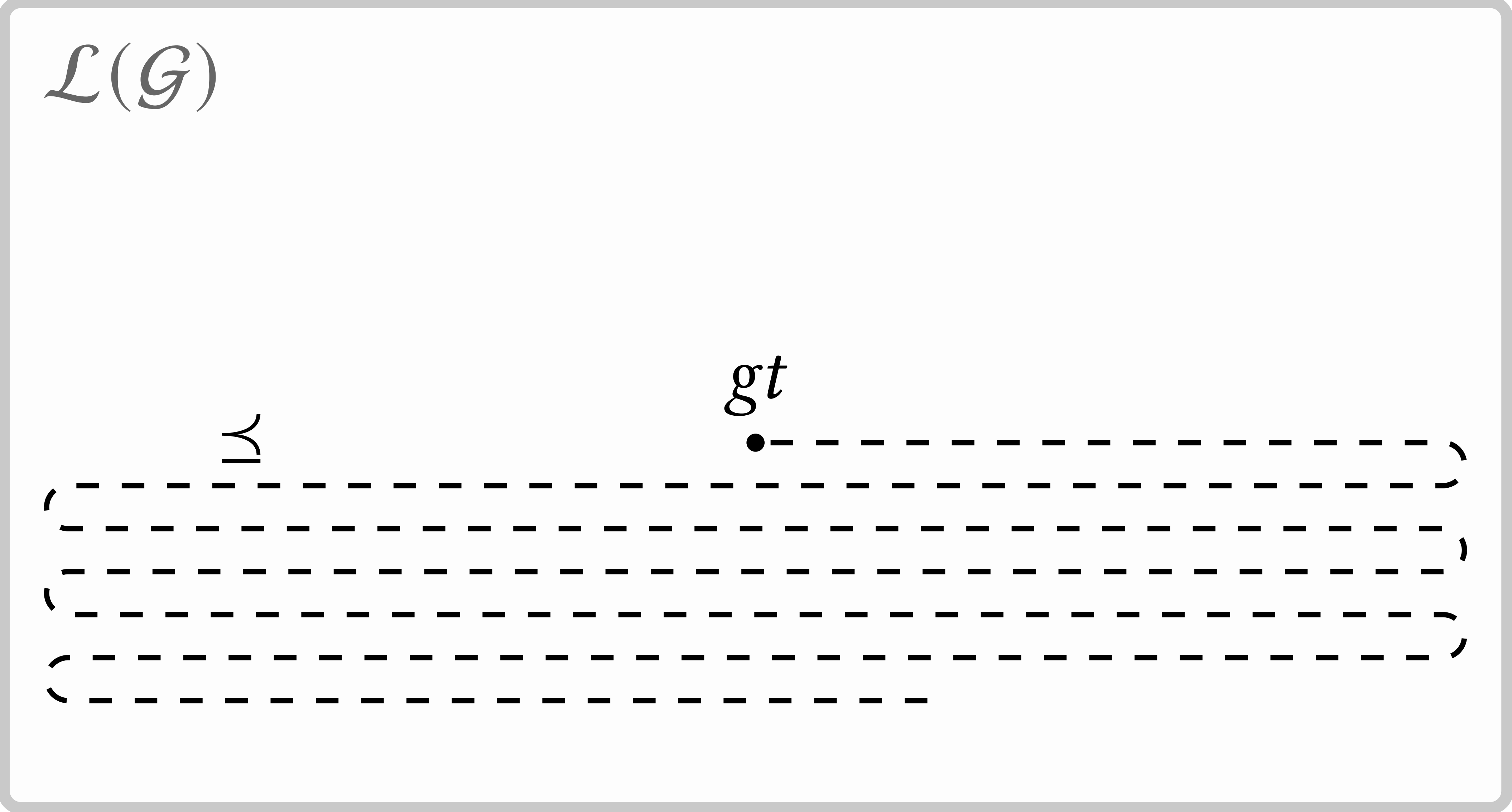
Pruning with a ball [Feser et al. 2023]



$\mathcal{L}(\mathcal{G})$
$\preceq$
gt
$\mathcal{L}(\mathcal{G})/_{\mathrm{OE}^{\#}}$

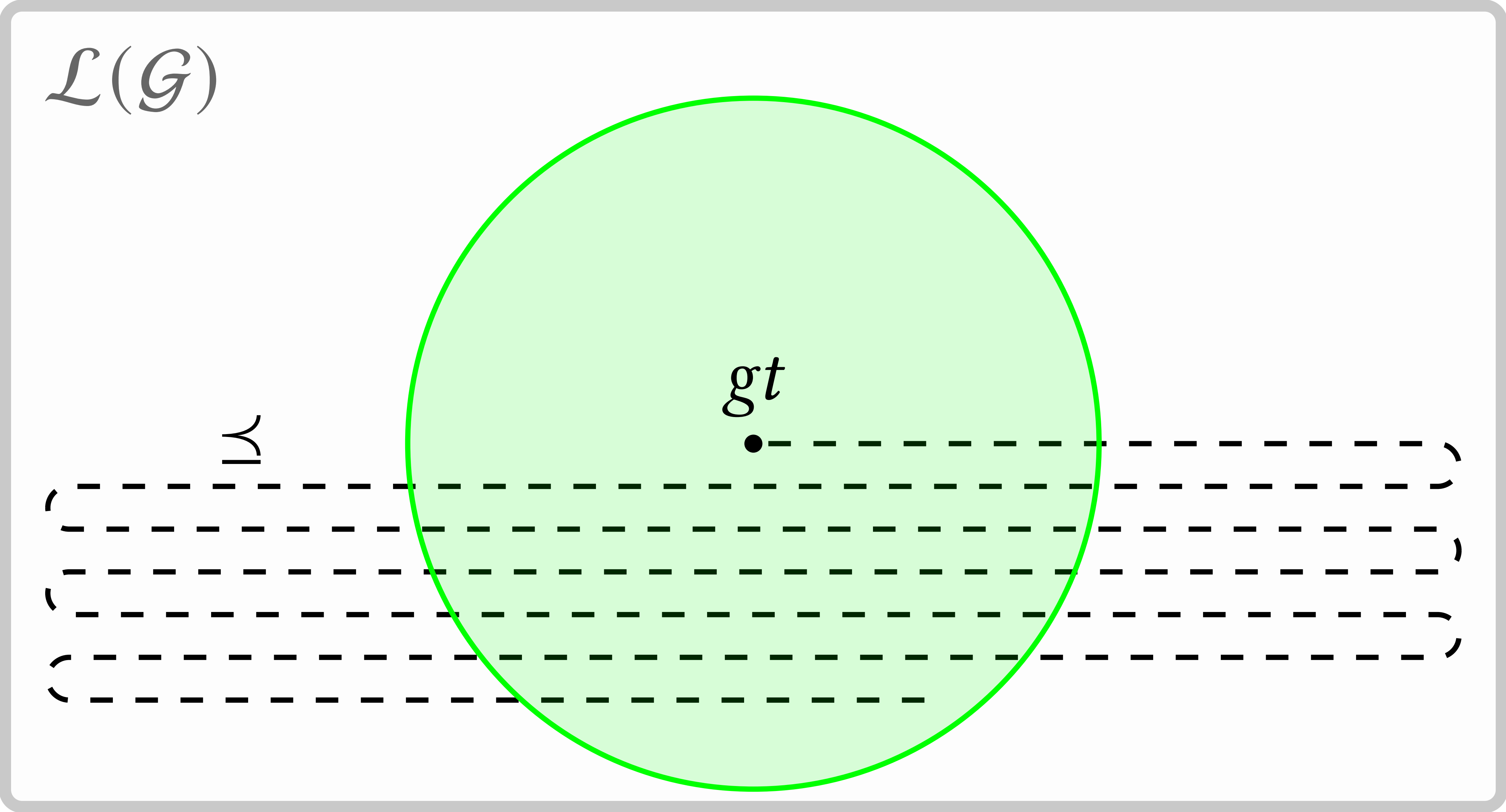
Pruning with a Ball

Use a metric to define a ball around gt



Pruning with a Ball

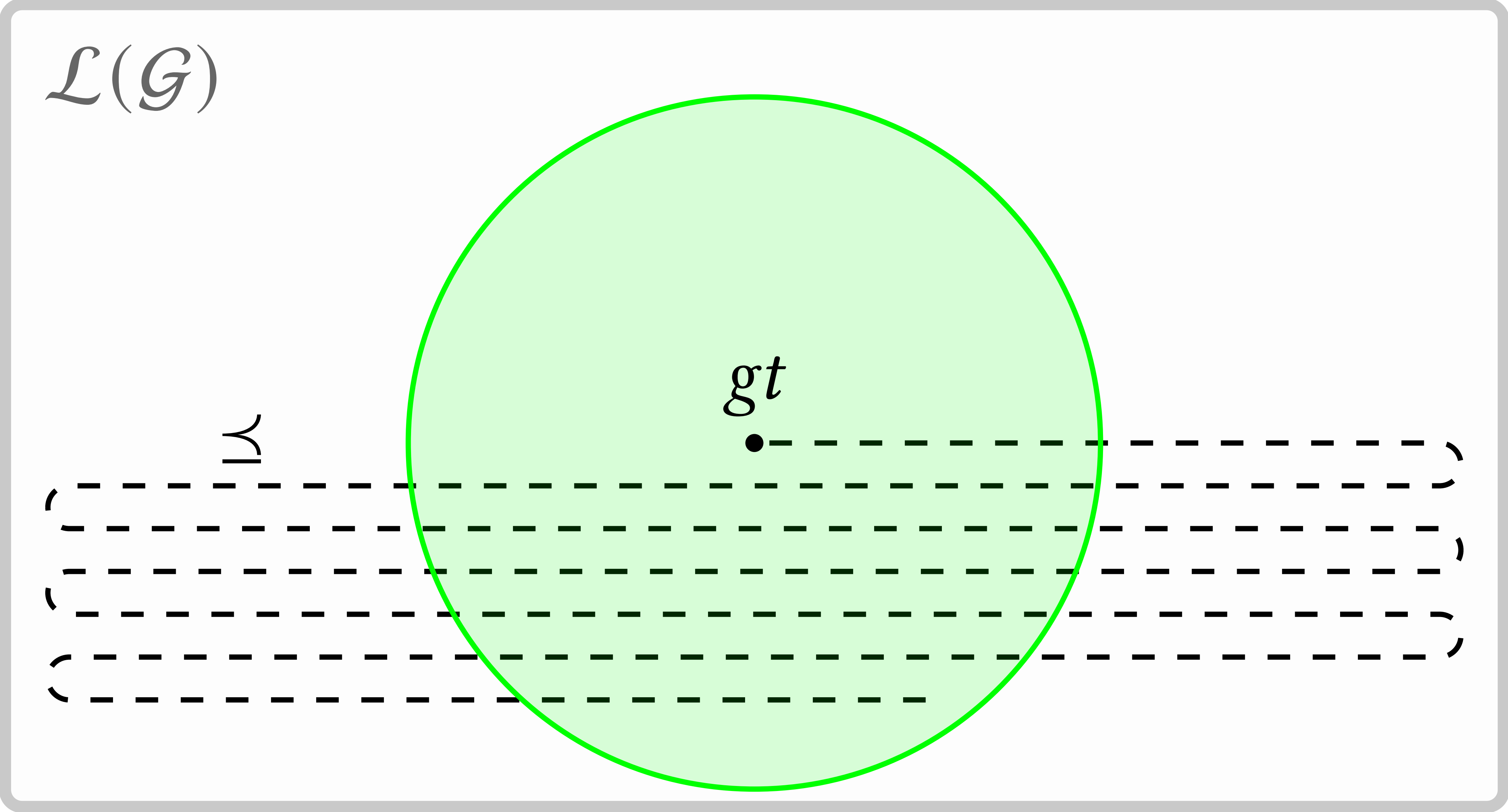
Use a metric to define a ball around gt



Pruning with a Ball

Use a metric to define a ball around gt

Only consider programs inside the ball

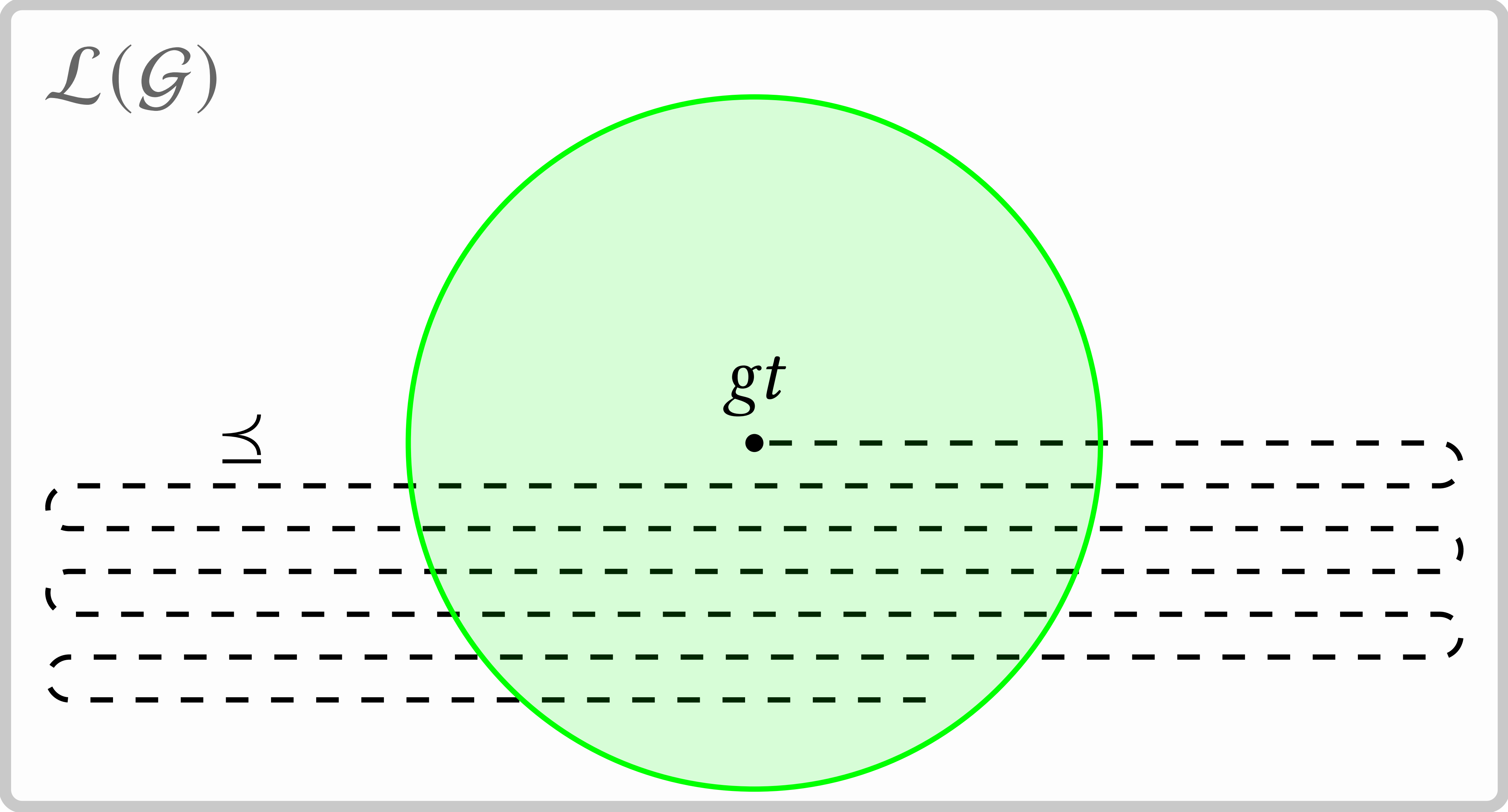


Pruning with a Ball

Use a metric to define a ball around gt

Only consider programs inside the ball

Deliberately incomplete



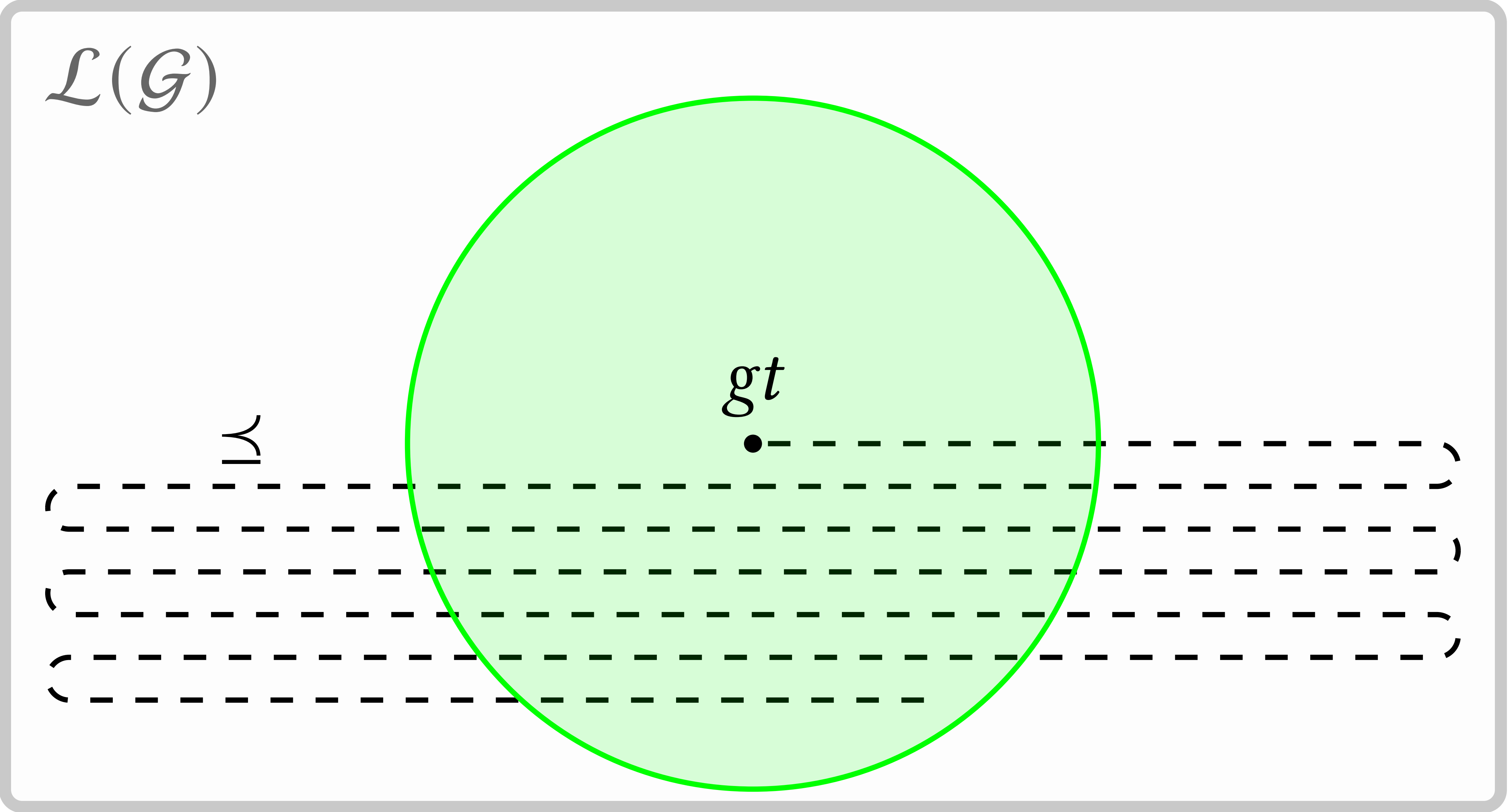
Pruning with a Ball

Use a metric to define a ball around gt

Only consider programs inside the ball

Deliberately incomplete

Control completeness / speed-up via radius



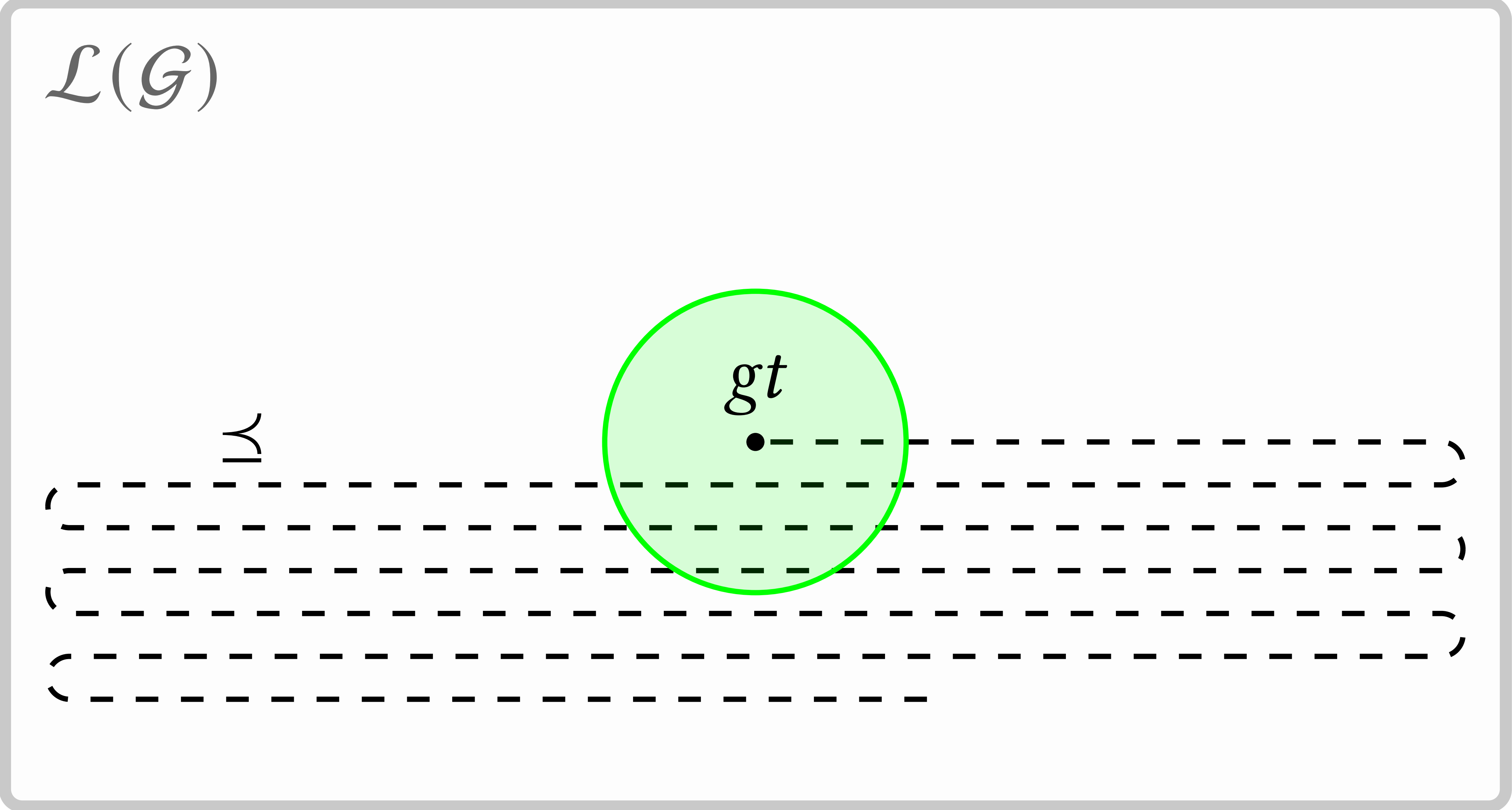
Pruning with a Ball

Use a metric to define a ball around gt

Only consider programs inside the ball

Deliberately incomplete

Control completeness / speed-up via radius



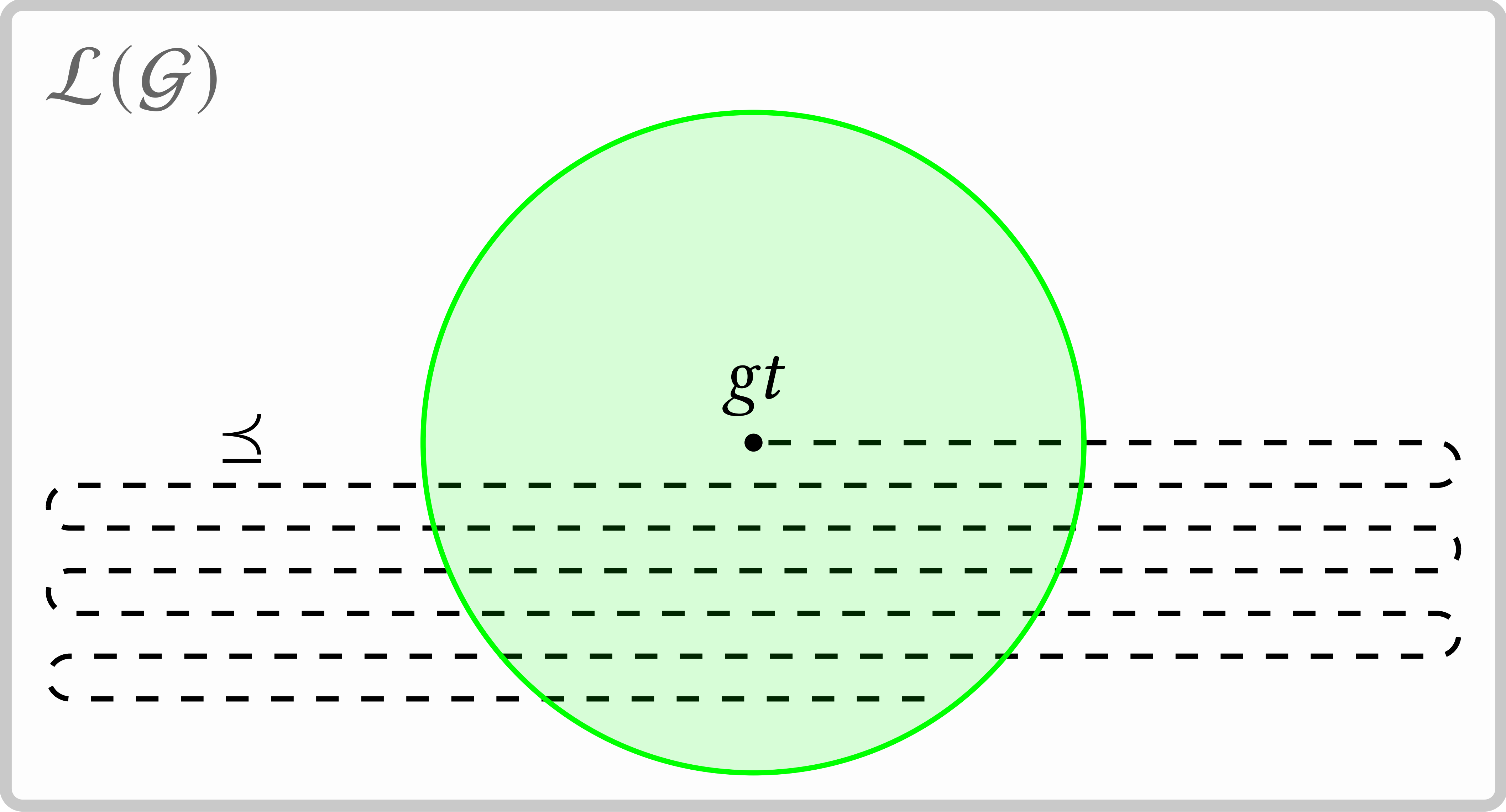
Pruning with a Ball

Use a metric to define a ball around gt

Only consider programs inside the ball

Deliberately incomplete

Control completeness / speed-up via radius



Pruning with a Ball

Use a metric to define a ball around gt

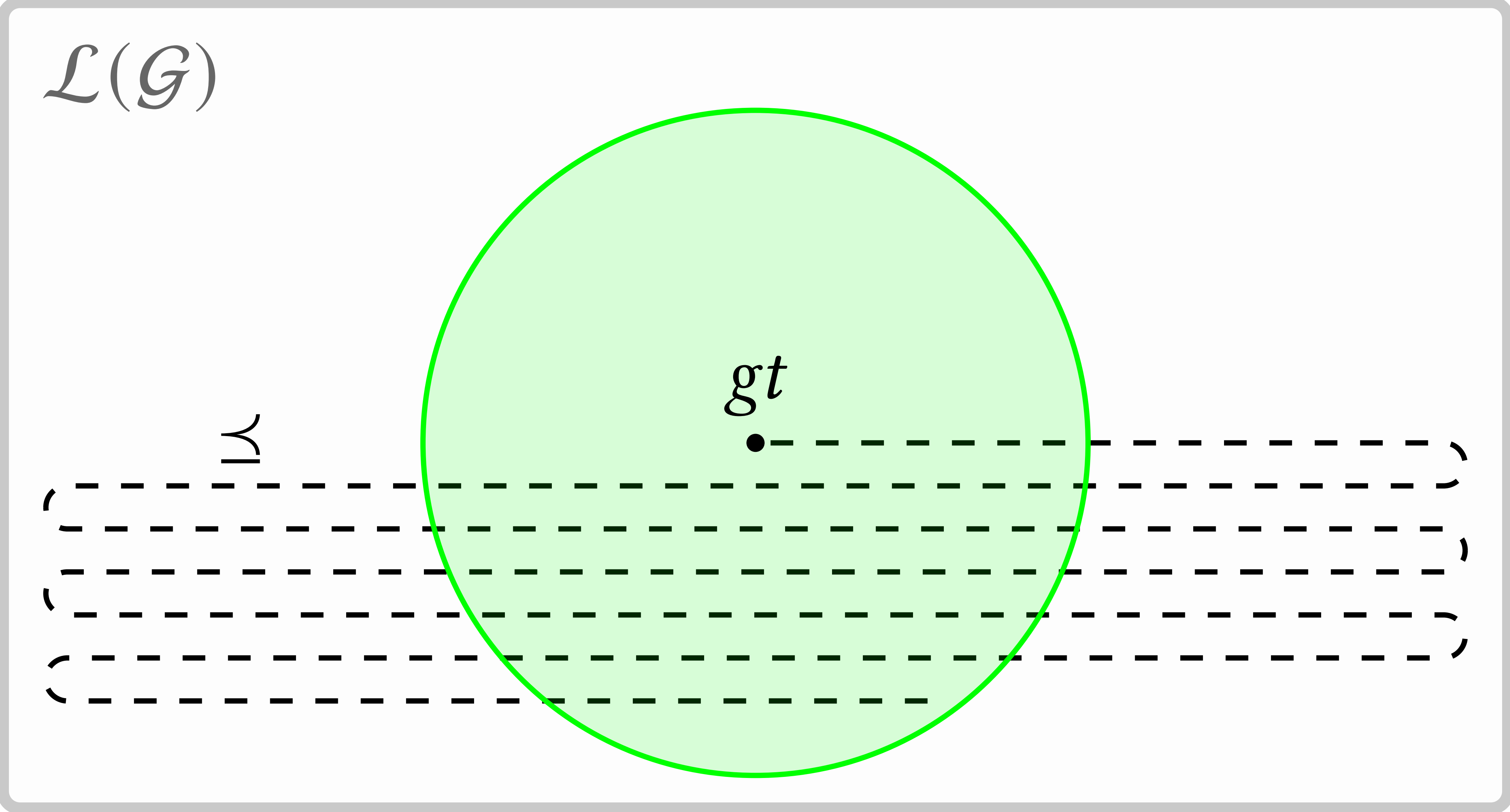
Only consider programs inside the ball

Deliberately incomplete

Control completeness / speed-up via radius

Downside:

- Require symmetry



Existing approaches

Existing approaches

have a way to **enumerate**

Existing approaches

have a way to **enumerate**

have a way to **factorize**

Existing approaches

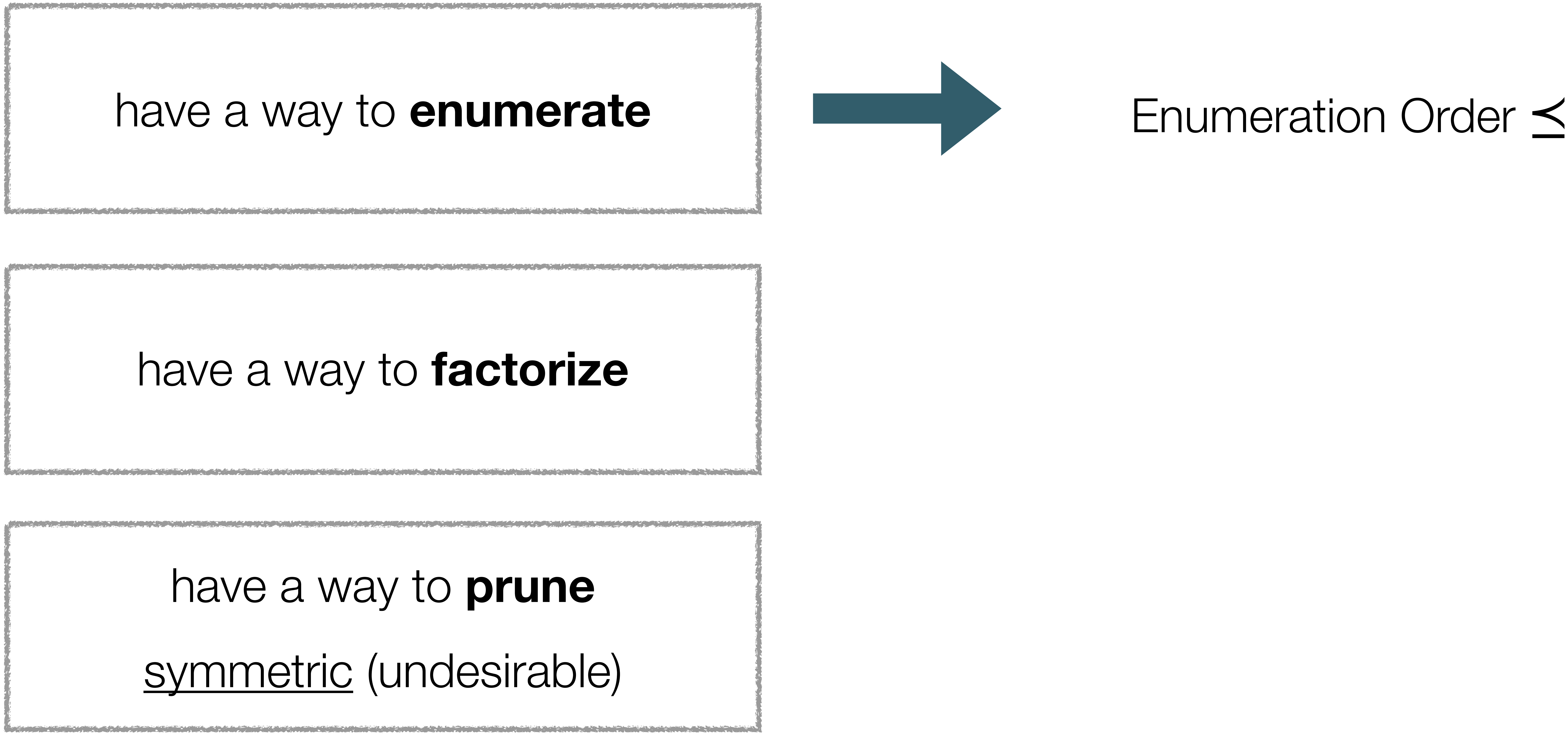
have a way to **enumerate**

have a way to **factorize**

have a way to **prune**

symmetric (undesirable)

Existing approaches



have a way to **enumerate**

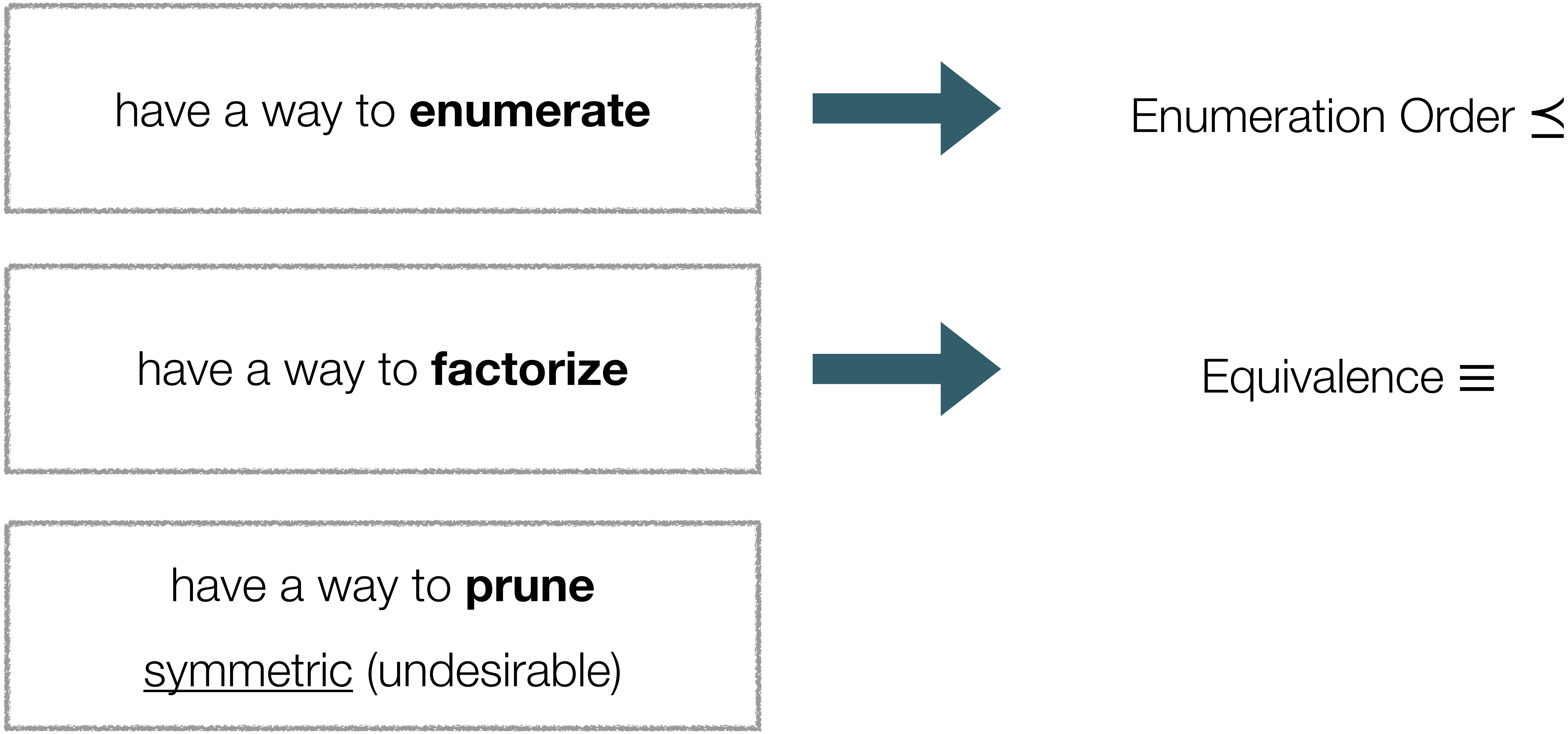
Enumeration Order $\preceq$

have a way to **factorize**

have a way to **prune**

symmetric (undesirable)

Existing approaches



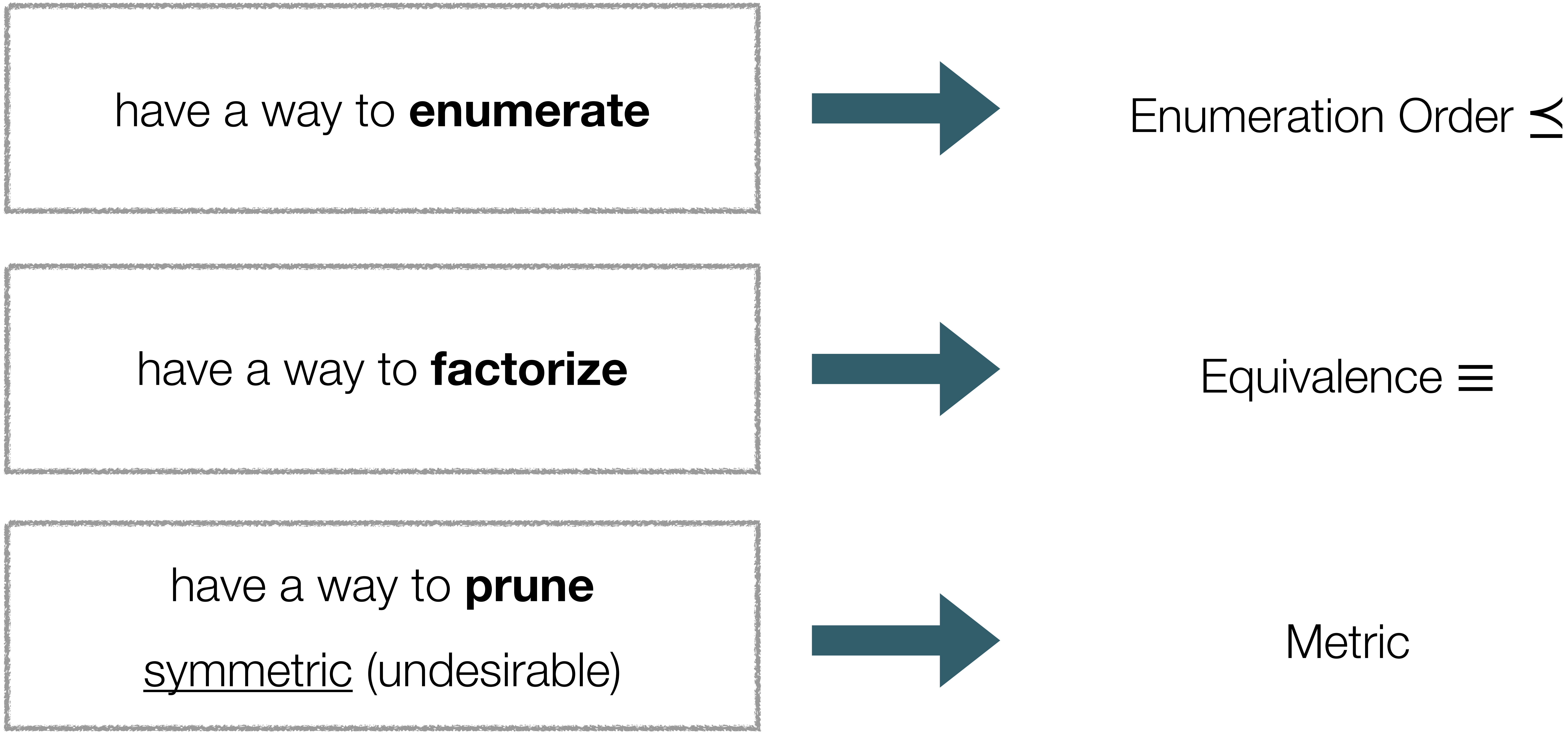
have a way to **enumerate** → Enumeration Order $\preceq$

have a way to **factorize** → Equivalence $\equiv$

have a way to **prune**

<u>symmetric</u> (undesirable)

Existing approaches



have a way to **enumerate** → Enumeration Order $\preceq$

have a way to **factorize** → Equivalence $\equiv$

have a way to **prune**

symmetric (undesirable) → Metric

3 Parameters of Bottom-Up Enumerative Synthesizers:

Enumeration Order $\preceq$

Equivalence $\equiv$

Metric

3 Parameters of Bottom-Up Enumerative Synthesizers:

Enumeration Order $\preceq$ $\underbrace{\text{Equivalence } \equiv \qquad \text{Metric}}$

INSIGHT: Oriented Metric

2 Parameters of Bottom-Up Enumerative Synthesizers:

Enumeration Order $\preceq$ $\underbrace{\cancel{\text{Equivalence} \equiv \qquad \text{Metric}}}$

INSIGHT: Oriented Metric

Oriented Metrics (Orimetrics)

$$m : D \times D \to \mathbb{R}_{\geq 0}$$

$$\begin{aligned} m(a,a) &= 0 && \text{(reflexivity)} \\ m(b,a) = 0 \implies m(a,b) &= 0 && \text{(symmetry at zero)} \\ m(a,c) \leq m(a,b) &+ m(b,c) && \text{($\triangle$-inequality)} \end{aligned}$$

Oriented Metrics (Orimetrics)

$$m : D \times D \to \mathbb{R}_{\geq 0}$$

$$m(a, a) = 0 \qquad \text{(reflexivity)}$$

$$m(b, a) = 0 \implies m(a, b) = 0 \qquad \text{(symmetry at zero)}$$

$$m(a, c) \leq m(a, b) + m(b, c) \qquad \text{(}\triangle\text{-inequality)}$$

Oriented Metrics (Orimetrics)

$$m : D \times D \longrightarrow \mathbb{R}_{\geq 0}$$

$$
\begin{aligned}
&& m(a,a) &= 0 && \text{(reflexivity)} \\
m(b,a) = 0 &\implies & m(a,b) &= 0 && \text{(symmetry at zero)} \\
m(a,c) &\leq & m(a,b) &+ m(b,c) && \text{($\triangle$-inequality)}
\end{aligned}
$$

Allows for asymmetry

Oriented Metrics (Orimetrics)

$$m : D \times D \longrightarrow \mathbb{R}_{\geq 0}$$

$$\begin{aligned} m(a,a) &= 0 && \text{(reflexivity)} \\ m(b,a) = 0 \;\Rightarrow\; m(a,b) &= 0 && \text{(symmetry at zero)} \\ m(a,c) &\leq m(a,b) + m(b,c) && \text{(}\triangle\text{-inequality)} \end{aligned}$$

Allows for asymmetry → Better pruning

Oriented Metrics (Orimetrics)

$$m : D \times D \to \mathbb{R}_{\geq 0}$$

$$\begin{aligned} && m(a, a) &= 0 && \text{(reflexivity)} \\ m(b, a) = 0 &\implies & m(a, b) &= 0 && \text{(symmetry at zero)} \\ m(a, c) &\leq & m(a, b) &+ m(b, c) && \text{($\triangle$-inequality)} \end{aligned}$$

Allows for asymmetry → Better pruning

Induces an equivalence

Oriented Metrics (Orimetrics)

$$m : D \times D \longrightarrow \mathbb{R}_{\geq 0}$$

$$\begin{aligned} m(a, a) &= 0 && \text{(reflexivity)} \\ m(b, a) = 0 \implies m(a, b) &= 0 && \text{(symmetry at zero)} \\ m(a, c) \leq m(a, b) &+ m(b, c) && \text{($\triangle$-inequality)} \end{aligned}$$

Allows for asymmetry → Better pruning

Induces an equivalence → OE factorization, abstraction

Why asymmetry?

SyGuS operators exhibit **asymmetric behavior**

Asymmetric Behavior: $\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$

Asymmetric Behavior: $\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\texttt{"POPL"}, \texttt{"PO"}) = m(\texttt{"PO"}, \texttt{"POPL"})$

Asymmetric Behavior: $\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\texttt{"POPL"}, \texttt{"PO"}) = m(\texttt{"PO"}, \texttt{"POPL"})$

$\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$ produces superstrings of $\mathcal{S}_1$ and $\mathcal{S}_2$

Asymmetric Behavior: $\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\texttt{"POPL"}, \texttt{"PO"}) = m(\texttt{"PO"}, \texttt{"POPL"})$

$\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$ produces superstrings of $\mathcal{S}_1$ and $\mathcal{S}_2$

`"POPL"` is a superstring of `"PO"`

Asymmetric Behavior: $\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\texttt{"POPL"}, \texttt{"PO"}) = m(\texttt{"PO"}, \texttt{"POPL"})$

$\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$ produces superstrings of $\mathcal{S}_1$ and $\mathcal{S}_2$

$\texttt{"POPL"}$ is a superstring of $\texttt{"PO"}$

$\texttt{"POPL"}$ cannot produce $\texttt{"PO"}$ with $\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$

Asymmetric Behavior: $\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\texttt{"POPL"}, \texttt{"PO"}) = m(\texttt{"PO"}, \texttt{"POPL"})$

$\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$ produces superstrings of $\mathcal{S}_1$ and $\mathcal{S}_2$

`"POPL"` is a superstring of `"PO"`

`"POPL"` cannot produce `"PO"` with $\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$

`"PO"` is a substring of `"POPL"`

Asymmetric Behavior: $\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\texttt{"POPL"}, \texttt{"PO"}) = m(\texttt{"PO"}, \texttt{"POPL"})$

$\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$ produces superstrings of $\mathcal{S}_1$ and $\mathcal{S}_2$

`"POPL"` is a superstring of `"PO"`

`"POPL"` cannot produce `"PO"` with $\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$

`"PO"` is a substring of `"POPL"`

`"PO"` might help produce `"POPL"` with $\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$

Asymmetric Behavior: $\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\texttt{"POPL"}, \texttt{"PO"}) = m(\texttt{"PO"}, \texttt{"POPL"})$

big small

$\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$ produces superstrings of $\mathcal{S}_1$ and $\mathcal{S}_2$

`"POPL"` is a superstring of `"PO"`

`"POPL"` cannot produce `"PO"` with $\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$

`"PO"` is a substring of `"POPL"`

`"PO"` might help produce `"POPL"` with $\texttt{concat}(\mathcal{S}_1, \mathcal{S}_2)$

Asymmetric Behavior: $\texttt{replace}(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$

Symmetry requires $m(\texttt{"POPL"}, \texttt{"PO"}) = m(\texttt{"PO"}, \texttt{"POPL"})$

Asymmetric Behavior: $\texttt{replace}(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$

Symmetry requires $m(\texttt{"POPL"}, \texttt{"PO"}) = m(\texttt{"PO"}, \texttt{"POPL"})$

$\texttt{replace}(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$ produces substrings of $\mathcal{S}_1$

Asymmetric Behavior: $\texttt{replace}(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$

Symmetry requires $m(\texttt{"POPL"}, \texttt{"PO"}) = m(\texttt{"PO"}, \texttt{"POPL"})$

$\texttt{replace}(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$ produces substrings of $\mathcal{S}_1$

`"POPL"` is a superstring of `"PO"`

`"POPL"` might help produce `"PO"` with $\texttt{replace}(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$

Asymmetric Behavior: `replace`$(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$

Symmetry requires $m($`"POPL"`$,$ `"PO"`$) = m($`"PO"`$,$ `"POPL"`$)$

`replace`$(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$ produces substrings of $\mathcal{S}_1$

`"POPL"` is a superstring of `"PO"`

`"POPL"` might help produce `"PO"` with `replace`$(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$

`"PO"` is a substring of `"POPL"`

`"PO"` cannot produce `"POPL"` with `replace`$(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$

Asymmetric Behavior: $\texttt{replace}(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$

Symmetry requires $m(\texttt{"POPL"}, \texttt{"PO"}) = m(\texttt{"PO"}, \texttt{"POPL"})$

<u>small</u> <u>big</u>

$\texttt{replace}(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$ produces substrings of $\mathcal{S}_1$

"POPL" is a superstring of "PO"

"POPL" might help produce "PO" with $\texttt{replace}(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$

"PO" is a substring of "POPL"

"PO" cannot produce "POPL" with $\texttt{replace}(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$

Asymmetric Behavior: $\mathsf{and}(\mathcal{S}_1, \mathcal{S}_2)$

Asymmetric Behavior: $\mathsf{and}(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\texttt{110}, \texttt{100}) = m(\texttt{100}, \texttt{110})$

Asymmetric Behavior: $\mathtt{and}(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\mathtt{110}, \mathtt{100}) = m(\mathtt{100}, \mathtt{110})$

$\mathtt{and}(\mathcal{S}_1, \mathcal{S}_2)$ produces bitvectors bitwise less than $\mathcal{S}_1$ and $\mathcal{S}_2$

Asymmetric Behavior: $\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\texttt{110}, \texttt{100}) = m(\texttt{100}, \texttt{110})$

$\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$ produces bitvectors bitwise less than $\mathcal{S}_1$ and $\mathcal{S}_2$

`110` is bitwise greater than `100`

Asymmetric Behavior: $\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\texttt{110}, \texttt{100}) = m(\texttt{100}, \texttt{110})$

$\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$ produces bitvectors bitwise less than $\mathcal{S}_1$ and $\mathcal{S}_2$

`110` is bitwise greater than `100`

`110` might help produce `100` with $\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$

Asymmetric Behavior: and$(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\texttt{110}, \texttt{100}) = m(\texttt{100}, \texttt{110})$

and$(\mathcal{S}_1, \mathcal{S}_2)$ produces bitvectors bitwise less than $\mathcal{S}_1$ and $\mathcal{S}_2$

`110` is bitwise greater than `100`

`110` might help produce `100` with and$(\mathcal{S}_1, \mathcal{S}_2)$

`100` is bitwise less than `110`

Asymmetric Behavior: $\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\texttt{110}, \texttt{100}) = m(\texttt{100}, \texttt{110})$

$\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$ produces bitvectors bitwise less than $\mathcal{S}_1$ and $\mathcal{S}_2$

`110` is bitwise greater than `100`

`110` might help produce `100` with $\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$

`100` is bitwise less than `110`

`100` cannot produce `110` with $\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$

Asymmetric Behavior: $\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\texttt{110}, \texttt{100}) = m(\texttt{100}, \texttt{110})$

small big

$\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$ produces bitvectors bitwise less than $\mathcal{S}_1$ and $\mathcal{S}_2$

`110` is bitwise greater than `100`

`110` might help produce `100` with $\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$

`100` is bitwise less than `110`

`100` cannot produce `110` with $\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$

Asymmetric Behavior: $\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$

Symmetry requires $m(\texttt{110}, \texttt{100}) = m(\texttt{100}, \texttt{110})$

small big

$\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$ produces bitvectors bitwise less than $\mathcal{S}_1$ and $\mathcal{S}_2$

`110` is bitwise greater than `100`

`110` might help produce `100` with $\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$

`100` is bitwise less than `110`

`100` cannot produce `110` with $\texttt{and}(\mathcal{S}_1, \mathcal{S}_2)$

We need asymmetry

Oriented Metrics (Orimetrics)

$$m : D \times D \longrightarrow \mathbb{R}_{\geq 0}$$

$$\begin{aligned}
&& m(a, a) &= 0 && \text{(reflexivity)} \\
m(b, a) = 0 &\implies & m(a, b) &= 0 && \text{(symmetry at zero)} \\
& m(a, c) \leq & m(a, b) &+ m(b, c) && \text{($\triangle$-inequality)}
\end{aligned}$$

Allows for asymmetry → Better pruning

Induces an equivalence → OE factorization, abstraction

Oriented Metrics (Orimetrics)

$$m : D \times D \to \mathbb{R}_{\geq 0}$$

$$m(a, a) \;=\; 0 \qquad \text{(reflexivity)}$$

$$m(b, a) \;=\; 0 \;\;\Rightarrow\;\; m(a, b) \;=\; 0 \qquad \text{(symmetry at zero)}$$

$$m(a, c) \;\leq\; m(a, b) \;+\; m(b, c) \qquad (\triangle\text{-inequality})$$

Equivalence at distance 0: $\quad a \equiv_m b \quad$ if $\quad m(a, b) = 0$

Oriented Metrics (Orimetrics)

$$m : D \times D \longrightarrow \mathbb{R}_{\geq 0}$$

$$\begin{aligned} m(a,a) \;&=\; 0 && \text{(reflexivity)} \\ m(b,a) \;=\; 0 \;\;\Longrightarrow\;\; m(a,b) \;&=\; 0 && \text{(symmetry at zero)} \\ m(a,c) \;\;\leq\;\; m(a,b) \;&+\; m(b,c) && \text{($\triangle$-inequality)} \end{aligned}$$

Equivalence at distance 0: $a \equiv_m b$ if $m(a,b) = 0$

→ OE factorization, abstraction

Oriented Metrics (Orimetrics)

$$m : D \times D \longrightarrow \mathbb{R}_{\geq 0}$$

$$\begin{aligned}
&& m(a, a) &= 0 && \text{(reflexivity)} \\
m(b, a) = 0 &\implies & m(a, b) &= 0 && \text{(symmetry at zero)} \\
m(a, c) &\leq & m(a, b) &+ m(b, c) && \text{($\triangle$-inequality)}
\end{aligned}$$

Allows for asymmetry → Better pruning

Induces an equivalence → OE factorization, abstraction

How to design an orimetric?

1. Construct an orimetric m on the data domain
2. Lift m to programs

Oriented Metric for $\texttt{replace}(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$ $m : D \times D \to \mathbb{R}_{\geq 0}$

Reward superstrings

Oriented Metric for `replace`$(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$

$m : D \times D \to \mathbb{R}_{\geq 0}$

Reward superstrings

1. For strings i, o:

Oriented Metric for `replace`$(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$ $m : D \times D \longrightarrow \mathbb{R}_{\geq 0}$

Reward superstrings

1. For strings i, o: $$m(i, o) = \begin{cases} len(i) - len(o) & \text{if } i \text{ is a superstring of } o \\ 100 + |len(i) - len(o)| & \text{otherwise} \end{cases}$$

Oriented Metric for `replace`$(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$

$m : D \times D \longrightarrow \mathbb{R}_{\geq 0}$

Reward superstrings

1. For strings i, o: $m(i, o) = \begin{cases} len(i) - len(o) & \text{if } i \text{ is a superstring of } o \\ 100 + |len(i) - len(o)| & \text{otherwise} \end{cases}$

$$m(\text{"PO"}, \text{"POPL"}) = 102$$

Oriented Metric for $\texttt{replace}(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$

$m : D \times D \longrightarrow \mathbb{R}_{\geq 0}$

Reward superstrings

1. For strings i, o: $m(i, o) = \begin{cases} len(i) - len(o) & \text{if } i \text{ is a superstring of } o \\ 100 + |len(i) - len(o)| & \text{otherwise} \end{cases}$

$m(\text{"PO"}, \text{"POPL"}) = 102$

$m(\text{"POPL"}, \text{"PO"}) = 2$

Oriented Metric for `replace`$(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$

$m : D \times D \longrightarrow \mathbb{R}_{\geq 0}$

Reward superstrings

1. For strings i, o: $$m(i, o) = \begin{cases} len(i) - len(o) & \text{if } i \text{ is a superstring of } o \\ 100 + |len(i) - len(o)| & \text{otherwise} \end{cases}$$

$m(\text{"PO"}, \text{"POPL"}) = 102$

$m(\text{"POPL"}, \text{"PO"}) = 2$

2. For programs **p**, **q**:

Oriented Metric for `replace`$(\mathcal{S}_1, \mathcal{S}_2, \epsilon)$ $\quad m : D \times D \longrightarrow \mathbb{R}_{\geq 0}$

Reward superstrings

1. For strings i, o: $\quad m(i, o) \; = \begin{cases} len(i) - len(o) & \text{if } i \text{ is a superstring of } o \\ 100 + |len(i) - len(o)| & \text{otherwise} \end{cases}$

$m(\text{"PO"}, \text{"POPL"}) \; = \; 102$ $\qquad\qquad$ $m(\text{"POPL"}, \text{"PO"}) \; = \; 2$

2. For programs p, q: $\quad m_{In}(\mathsf{p}, \mathsf{q}) \; = \; \sum_{i \in In} m(\mathsf{p}(i), \mathsf{q}(i))$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

Pruning with an Orimetric

$$m(i, o) \;=\; \begin{cases} len(i) - len(o) & \text{if } i \text{ is a superstring of } o \\ 100 + |len(i) - len(o)| & \text{otherwise} \end{cases}$$

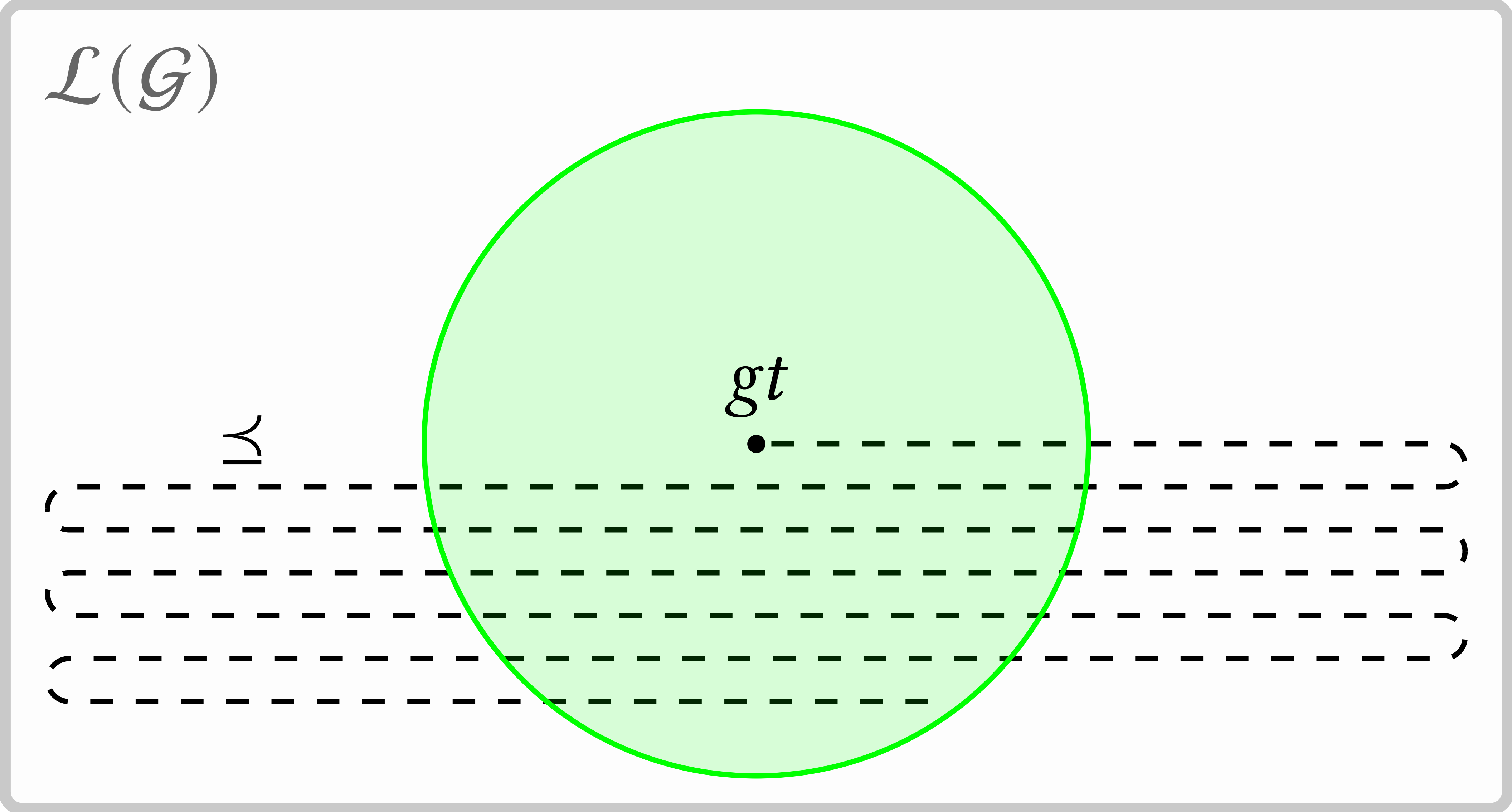
$$m_{In}(\mathsf{p}, \mathsf{q}) \;=\; \sum_{i \in In} m(\mathsf{p}(i), \mathsf{q}(i))$$

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

Pruning with an Orimetric

$$m(i,o) = \begin{cases} len(i) - len(o) & \text{if } i \text{ is a superstring of } o \\ 100 + |len(i) - len(o)| & \text{otherwise} \end{cases}$$

$$m_{In}(\mathsf{p},\mathsf{q}) = \sum_{i \in In} m(\mathsf{p}(i), \mathsf{q}(i))$$



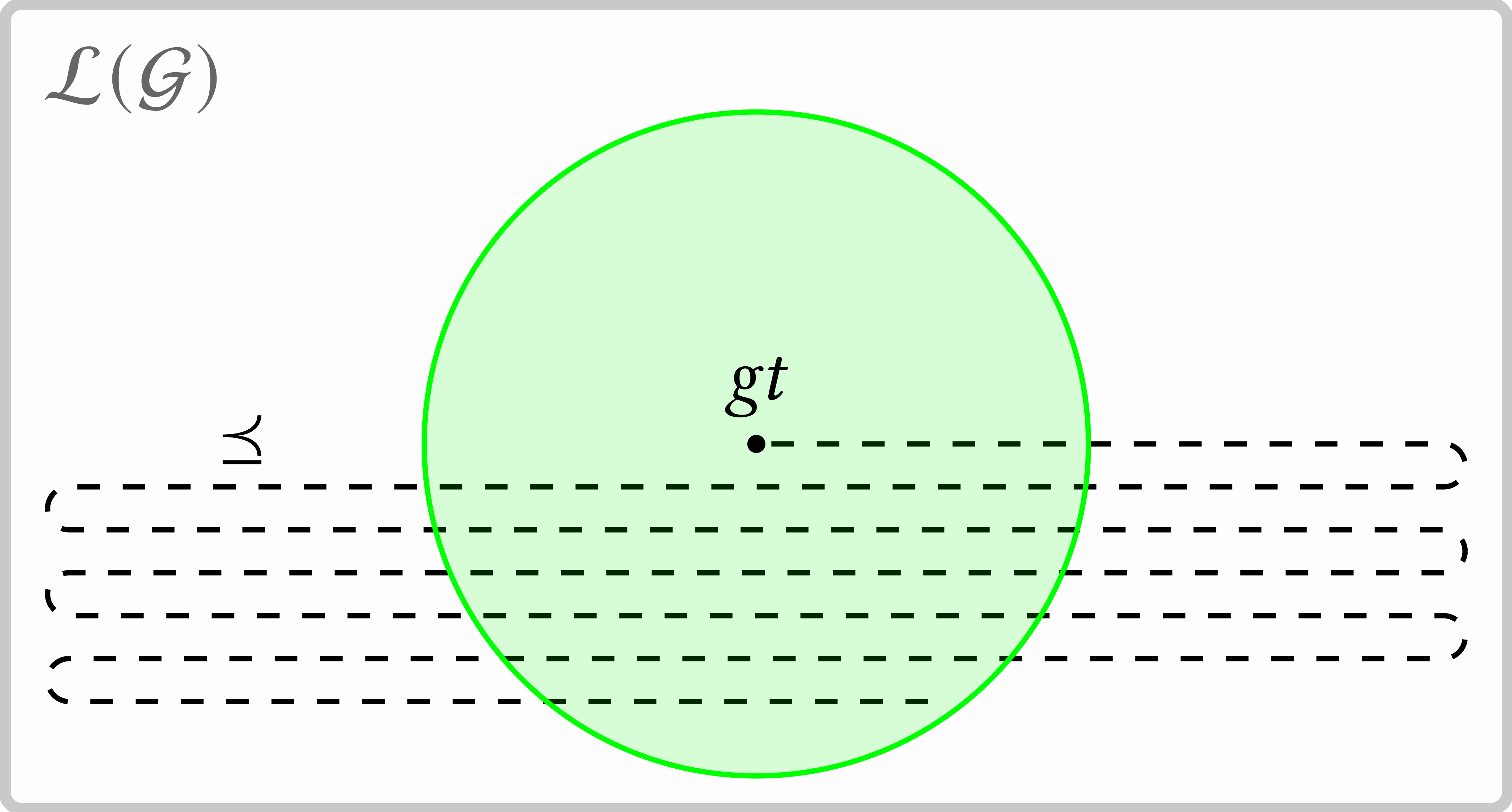
Set radius r to 100.

Pruning with an Orimetric

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

$$m(i, o) \;=\; \begin{cases} \mathit{len}(i) - \mathit{len}(o) & \text{if } i \text{ is a superstring of } o \\ 100 + |\mathit{len}(i) - \mathit{len}(o)| & \text{otherwise} \end{cases}$$

$$m_{In}(\mathsf{p}, \mathsf{q}) \;=\; \sum_{i \in In} m(\mathsf{p}(i), \mathsf{q}(i))$$



`"␣City"."␣City"`

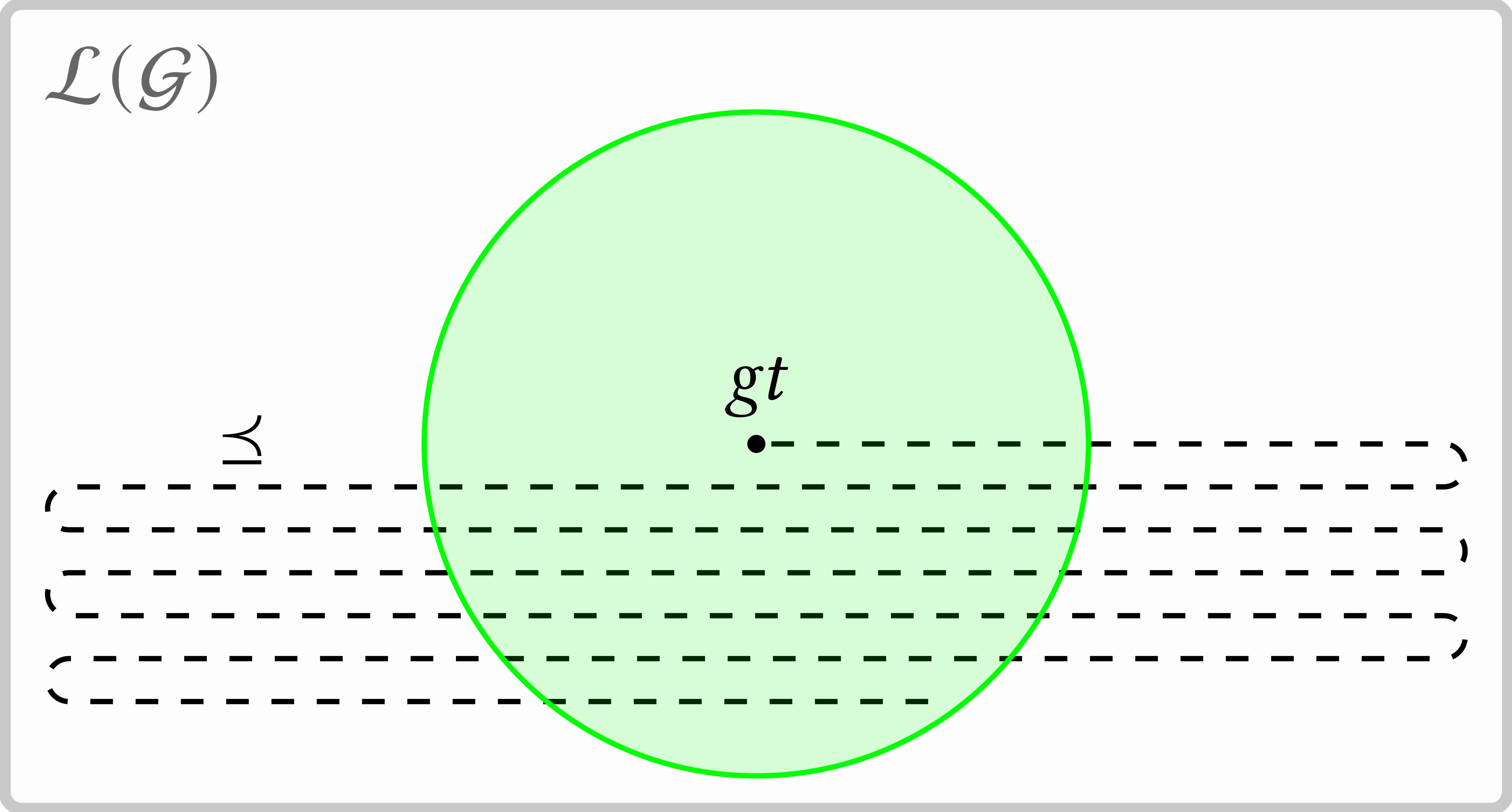
Set radius r to 100.

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

Pruning with an Orimetric

$$m(i, o) = \begin{cases} len(i) - len(o) & \text{if } i \text{ is a superstring of } o \\ 100 + |len(i) - len(o)| & \text{otherwise} \end{cases}$$

$$m_{In}(\mathsf{p}, \mathsf{q}) = \sum_{i \in In} m(\mathsf{p}(i), \mathsf{q}(i))$$



"␣City"."␣City"

$$m_{In}(\texttt{"␣City"."␣City"}, gt) > 100$$

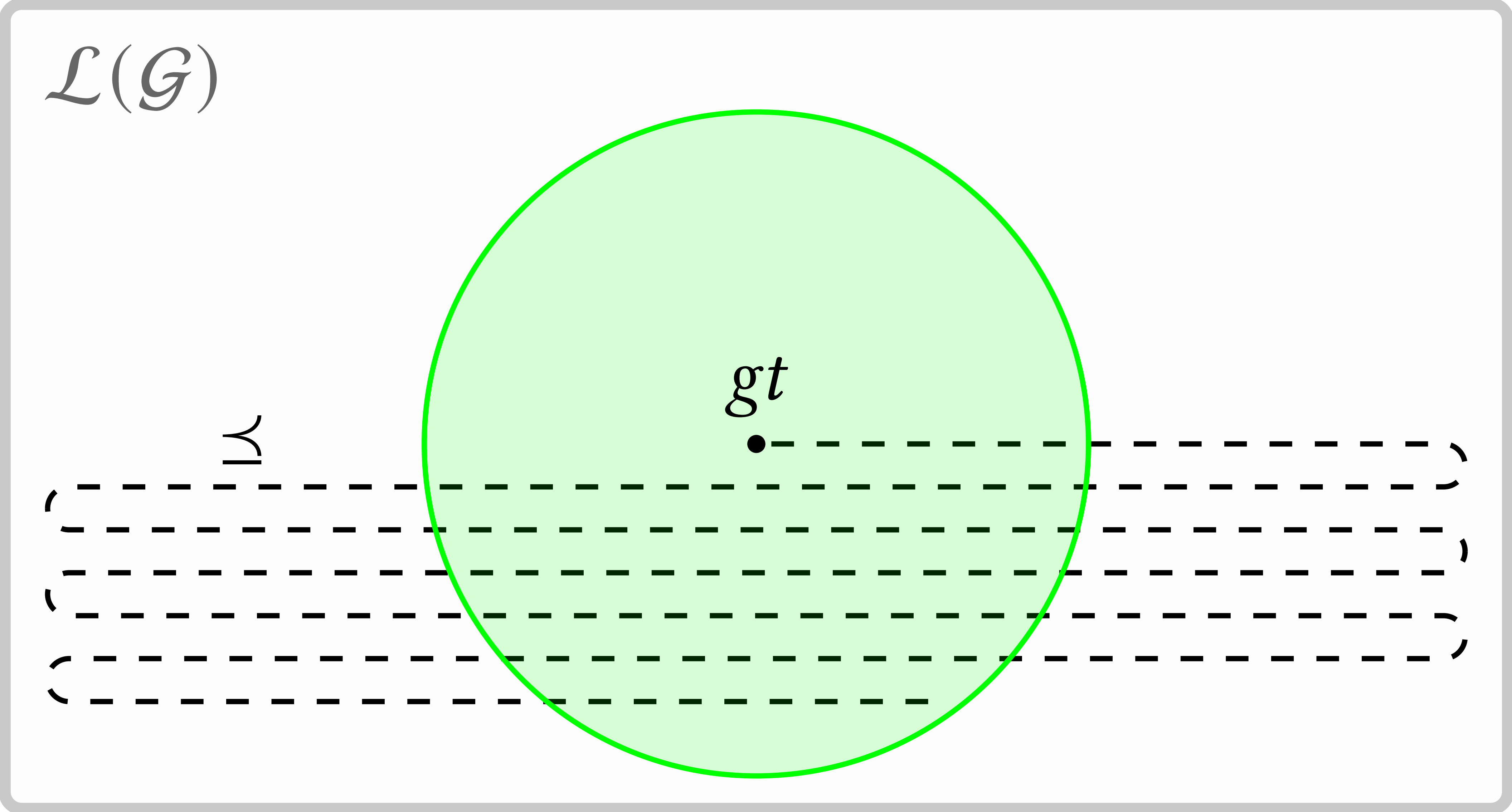
Set radius r to 100.

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

Pruning with an Orimetric

$$m(i, o) \;=\; \begin{cases} len(i) - len(o) & \text{if } i \text{ is a superstring of } o \\ 100 + |len(i) - len(o)| & \text{otherwise} \end{cases}$$

$$m_{In}(\mathsf{p}, \mathsf{q}) \;=\; \sum_{i \in In} m(\mathsf{p}(i), \mathsf{q}(i))$$



"␣City✗␣City"

$$m_{In}(\texttt{"␣City"}.\texttt{"␣City"}, gt) > 100$$

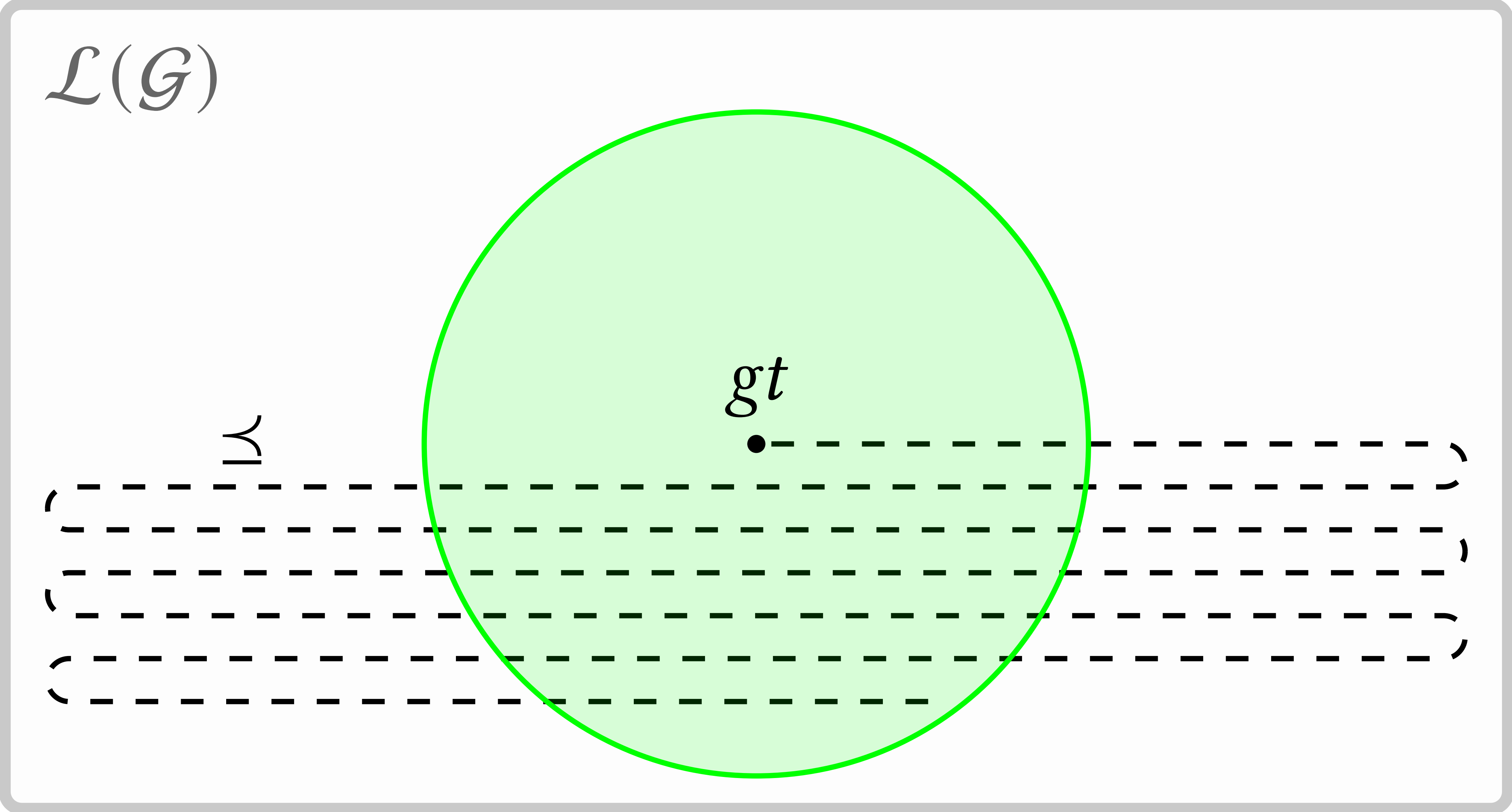
Set radius r to 100.

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

Pruning with an Orimetric

$$m(i, o) = \begin{cases} len(i) - len(o) & \text{if } i \text{ is a superstring of } o \\ 100 + |len(i) - len(o)| & \text{otherwise} \end{cases}$$

$$m_{In}(\mathsf{p}, \mathsf{q}) = \sum_{i \in In} m(\mathsf{p}(i), \mathsf{q}(i))$$



Method	P_1	P_2	P_3	P_4	P_5	P_6	P_7
No Pruning or Factorization	4	-	16	64	128	1280	4352
OE Factorization	4	-	9	6	27	56	119
Orimetric Pruning (OP)	4	-	7	18	56	323	929

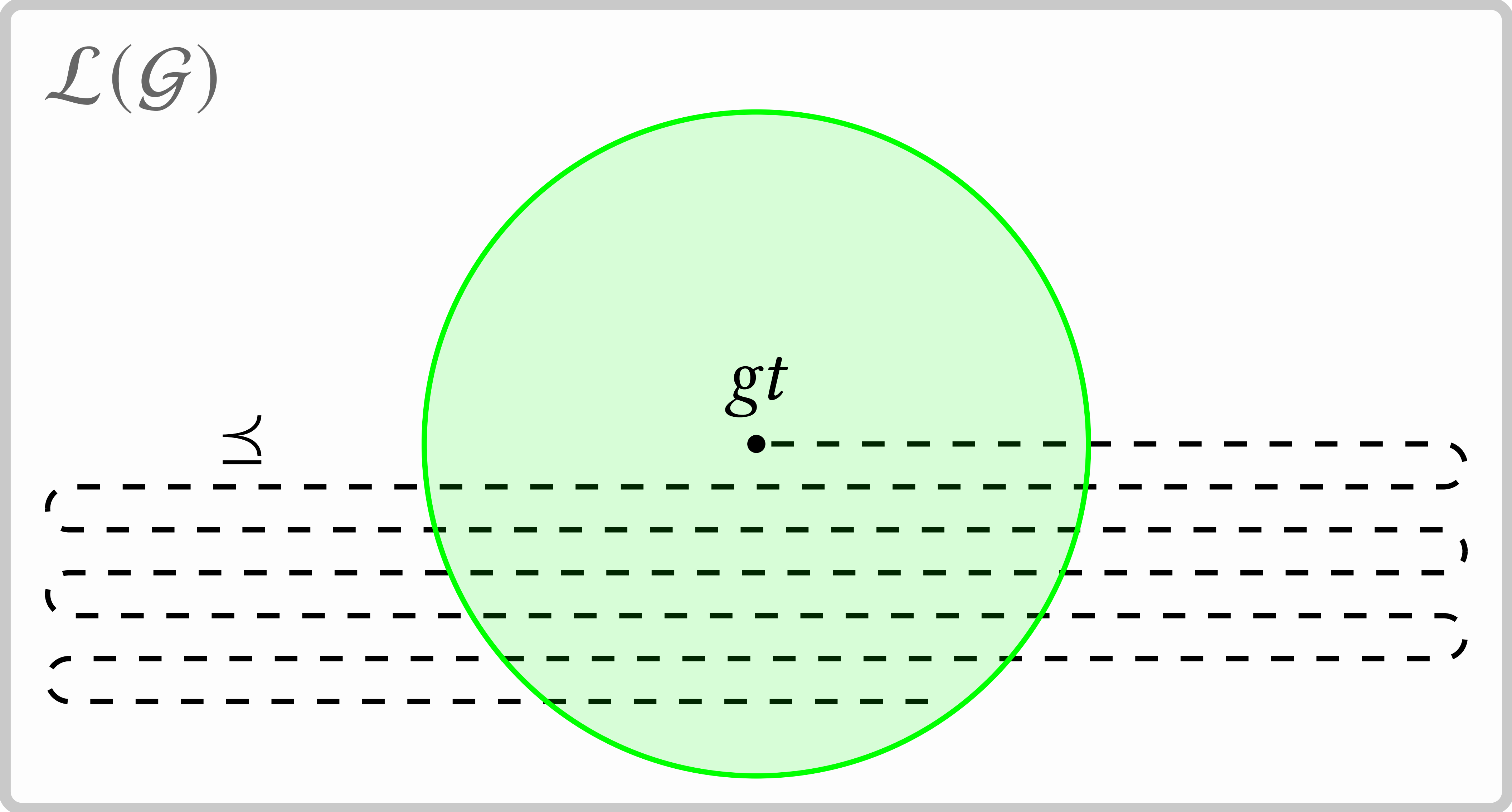
Set radius r to 100.

Factorizing with an Orimetric

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

$$m(i, o) \;=\; \begin{cases} len(i) - len(o) & \text{if } i \text{ is a superstring of } o \\ 100 + |len(i) - len(o)| & \text{otherwise} \end{cases}$$

$$m_{In}(\mathsf{p}, \mathsf{q}) \;=\; \sum_{i \in In} m(\mathsf{p}(i), \mathsf{q}(i))$$



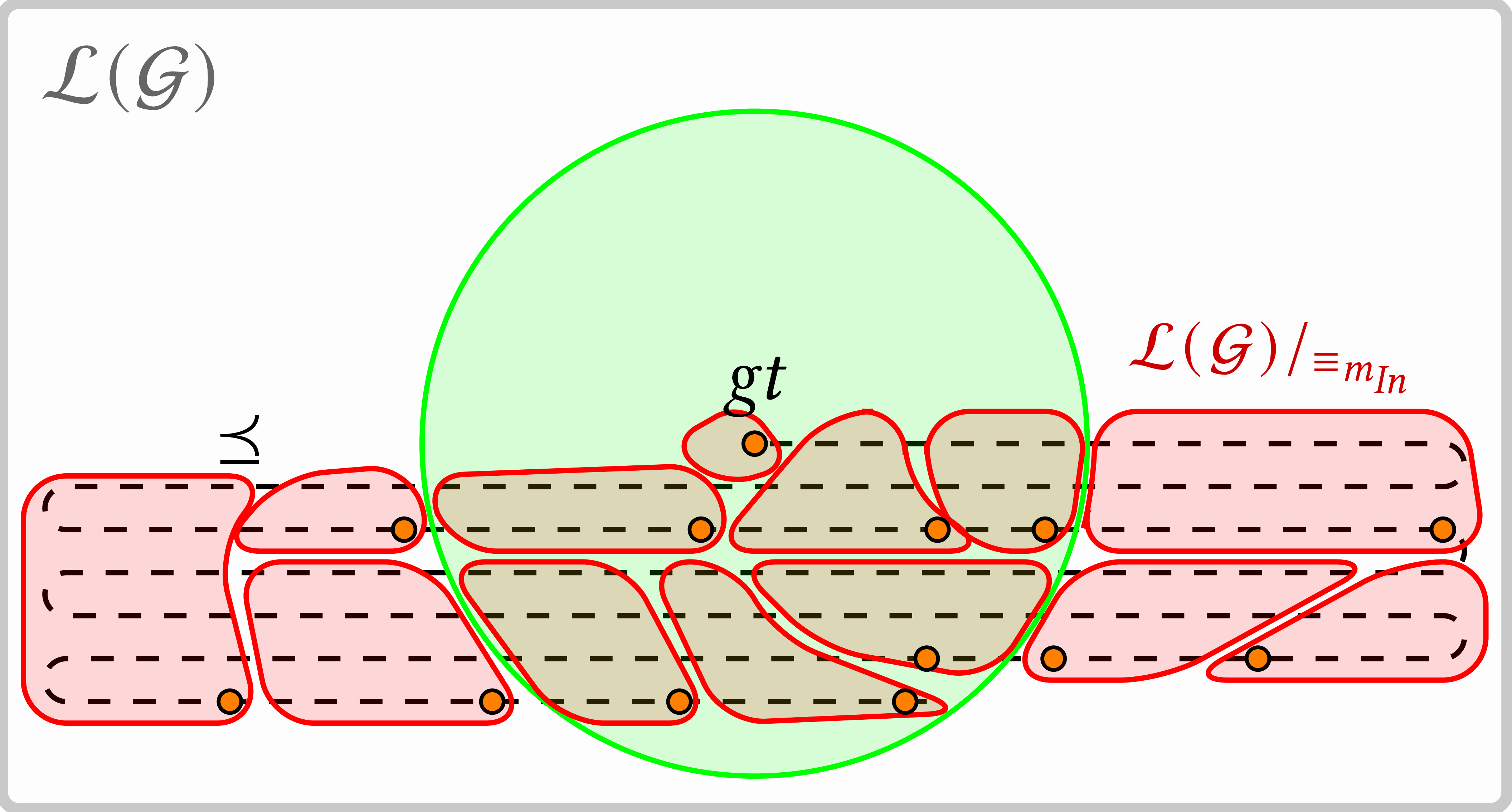
Method	P_1	P_2	P_3	P_4	P_5	P_6	P_7
No Pruning or Factorization	4	-	16	64	128	1280	4352
OE Factorization	4	-	9	6	27	56	119
Orimetric Pruning (OP)	4	-	7	18	56	323	929

Factorizing with an Orimetric

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

$$m(i, o) = \begin{cases} len(i) - len(o) & \text{if } i \text{ is a superstring of } o \\ 100 + |len(i) - len(o)| & \text{otherwise} \end{cases}$$

$$m_{In}(\mathsf{p}, \mathsf{q}) = \sum_{i \in In} m(\mathsf{p}(i), \mathsf{q}(i))$$



$\mathcal{L}(\mathcal{G})$
gt
$\preceq$
$\mathcal{L}(\mathcal{G})/_{\equiv_{m_{In}}}$

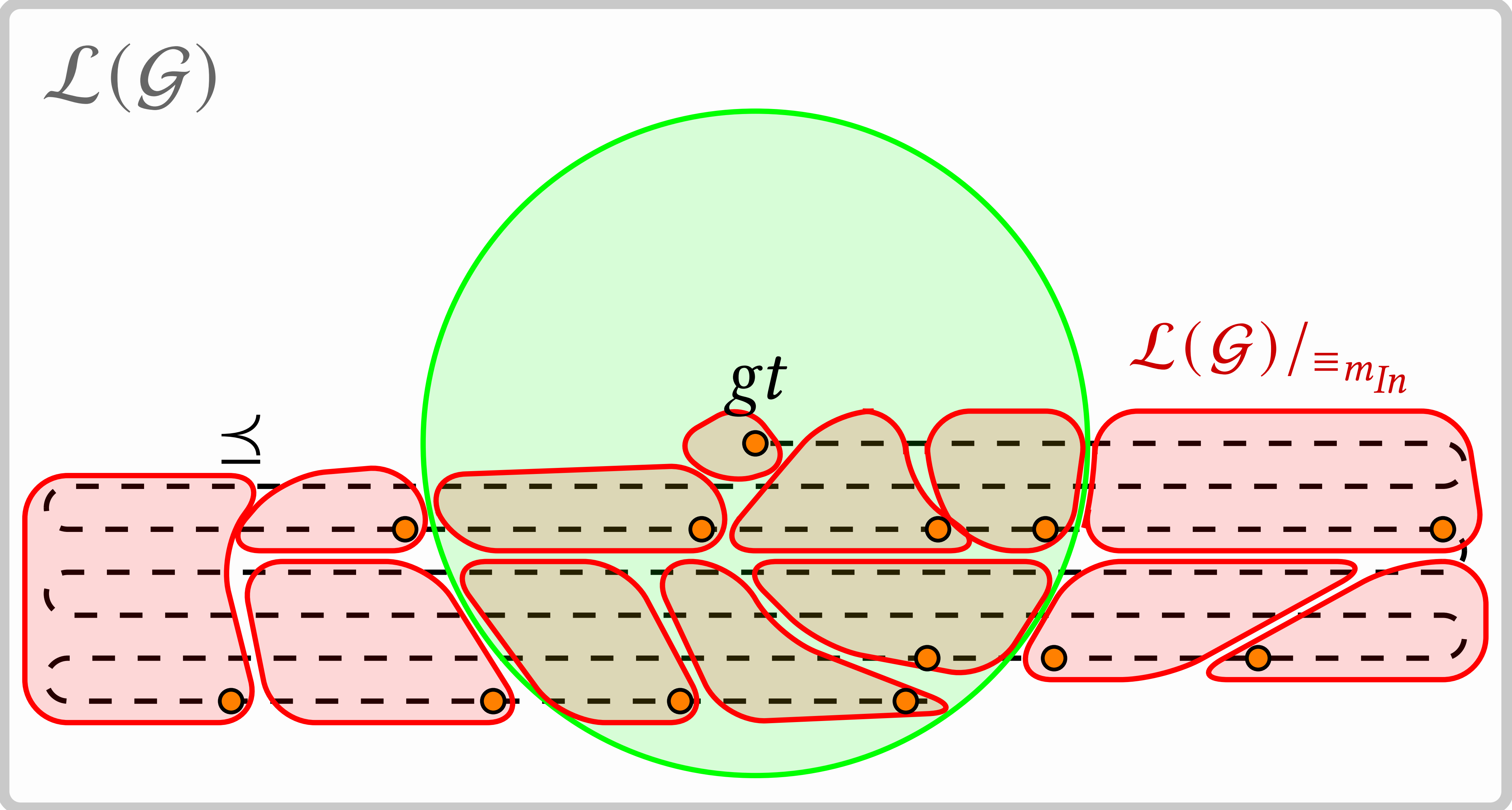
Method	P_1	P_2	P_3	P_4	P_5	P_6	P_7
No Pruning or Factorization	4	-	16	64	128	1280	4352
OE Factorization	4	-	9	6	27	56	119
Orimetric Pruning (OP)	4	-	7	18	56	323	929

In	*Out*
"POPL␣Conf"	"POPL"
"Rennes␣City"	"Rennes"
"PLDI␣Conf"	"PLDI"
"Seoul␣City"	"Seoul"

Factorizing with an Orimetric

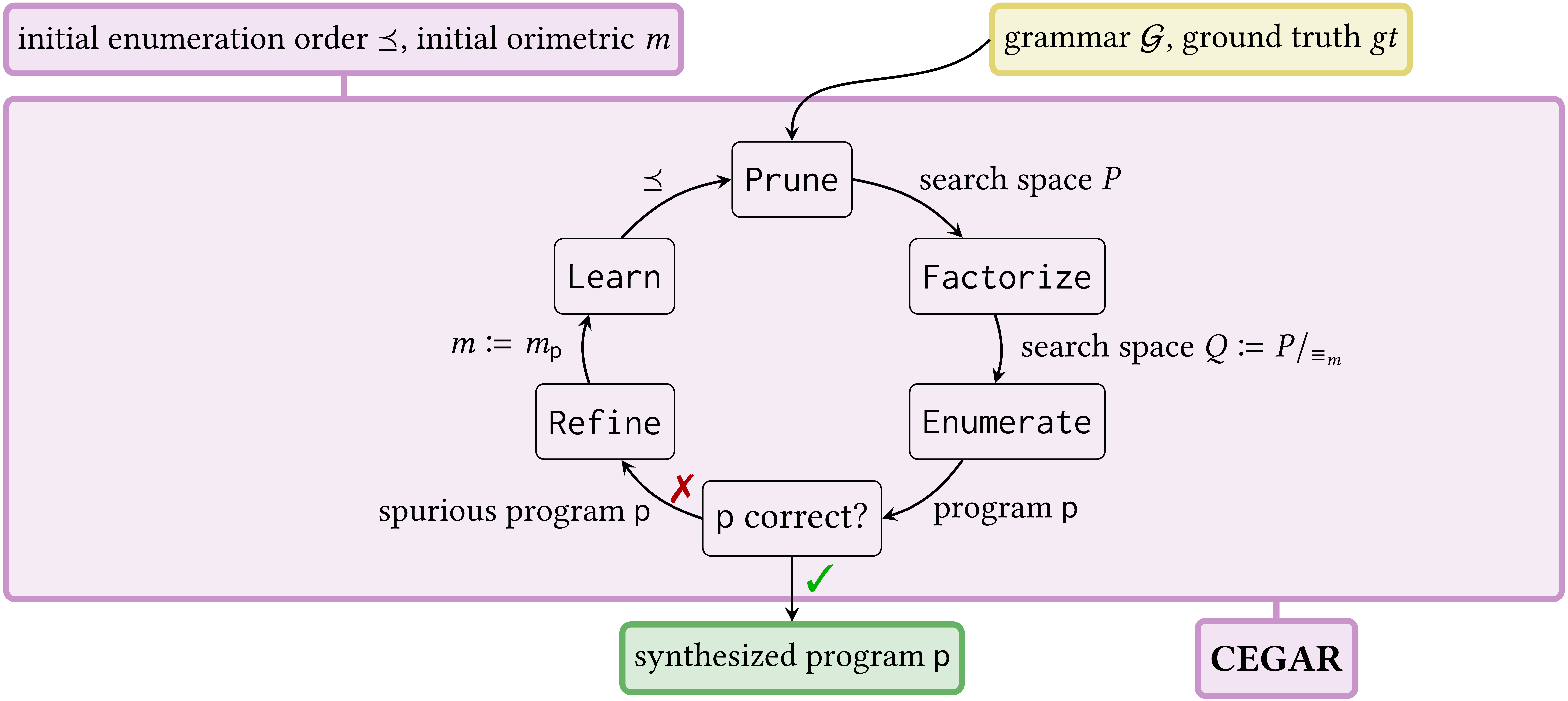
$$m(i, o) \;=\; \begin{cases} len(i) - len(o) & \text{if } i \text{ is a superstring of } o \\ 100 + |len(i) - len(o)| & \text{otherwise} \end{cases}$$

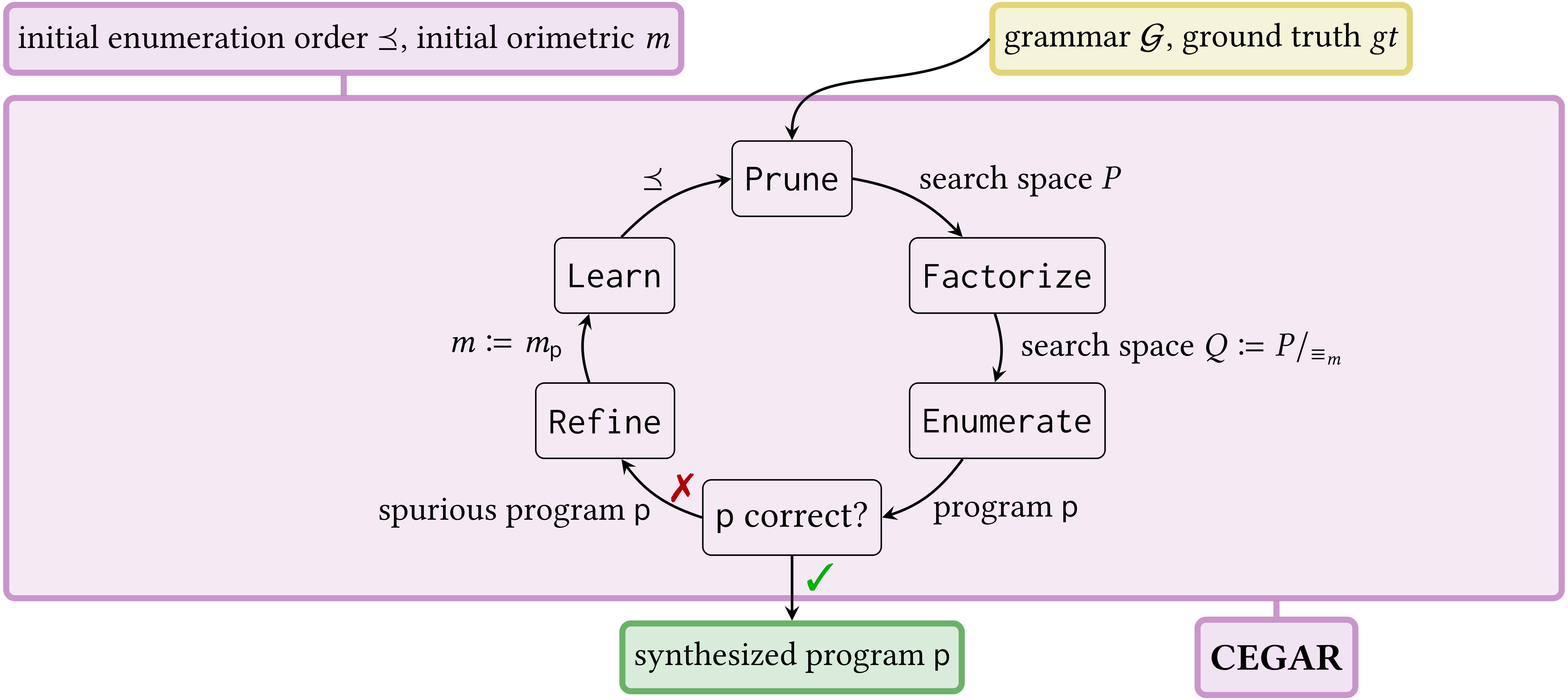
$$m_{In}(\mathsf{p}, \mathsf{q}) \;=\; \sum_{i \in In} m(\mathsf{p}(i), \mathsf{q}(i))$$



$\mathcal{L}(\mathcal{G})$
gt
$\preceq$
$\mathcal{L}(\mathcal{G})/_{\equiv_{m_{In}}}$

Method	P_1	P_2	P_3	P_4	P_5	P_6	P_7
No Pruning or Factorization	4	-	16	64	128	1280	4352
OE Factorization	4	-	9	6	27	56	119
Orimetric Pruning (OP)	4	-	7	18	56	323	929
OE Factorization + OP	4	-	5	6	19	50	81


initial enumeration order $\preceq$, initial orimetric m
grammar $\mathcal{G}$, ground truth gt
Prune
search space P
Factorize
search space $Q := P/_{\equiv_m}$
Enumerate
program p
p correct?
✓
synthesized program p
✗
spurious program p
Refine
$m := m_{\mathsf{p}}$
Learn
$\preceq$
CEGAR



In practice: concurrent instances employing different orimetrics

Evaluation of Merlin

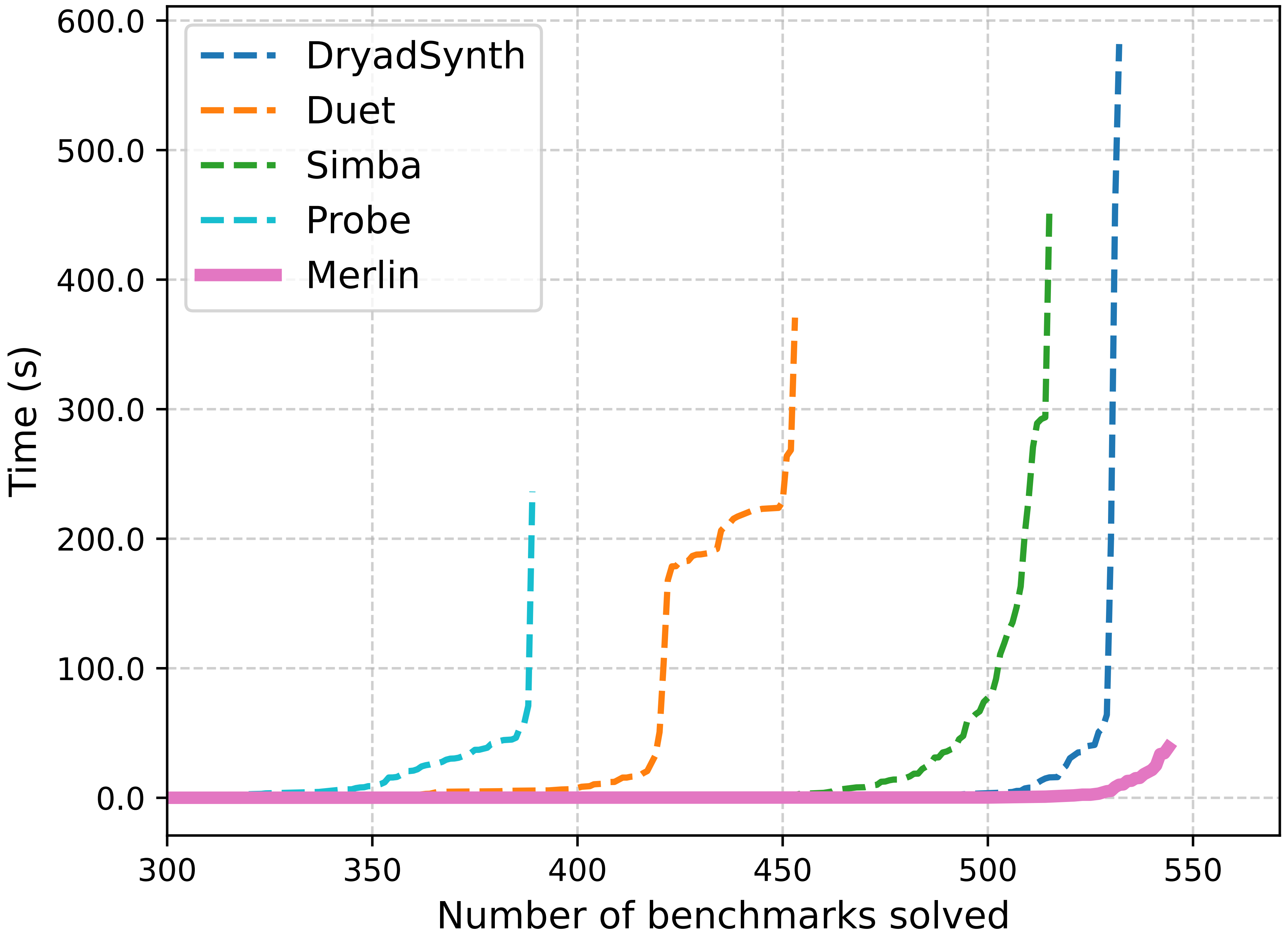
Artifacts Available acm V1.1

Artifacts Evaluated acm Reusable V1.1

Evaluation of Merlin

SyGuS-Bitvector

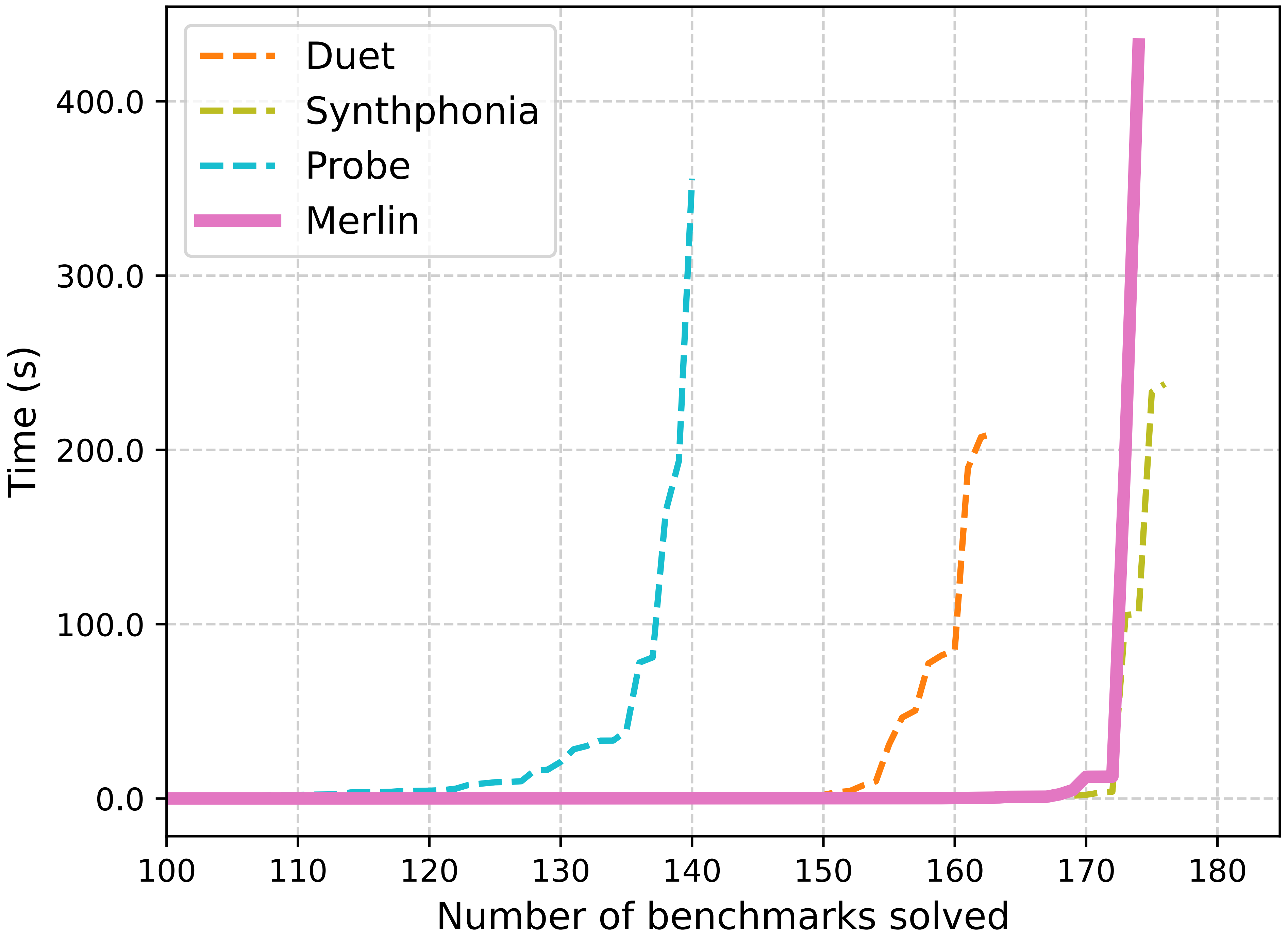
500 Deobfusc [Yoon et. al 2023]   49 Hacker's Delight [Warren 2013]



25x faster than DryadSynth

SyGuS-String

181 Duet [Lee 2021]



Comptetitive with Synthphonia

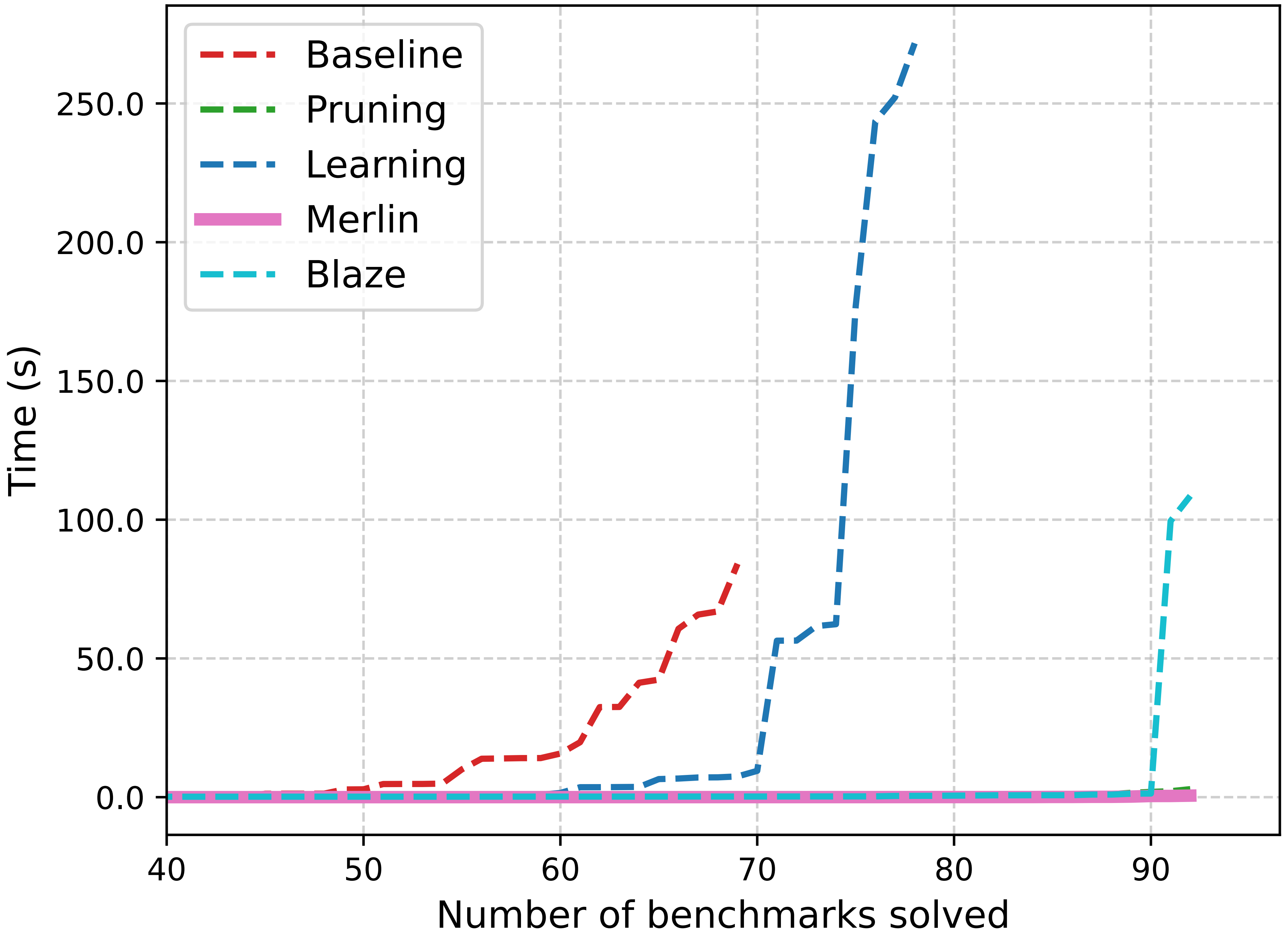
Evaluation of Merlin

Artifacts Available acm V1.1

Artifacts Evaluated acm Reusable V1.1

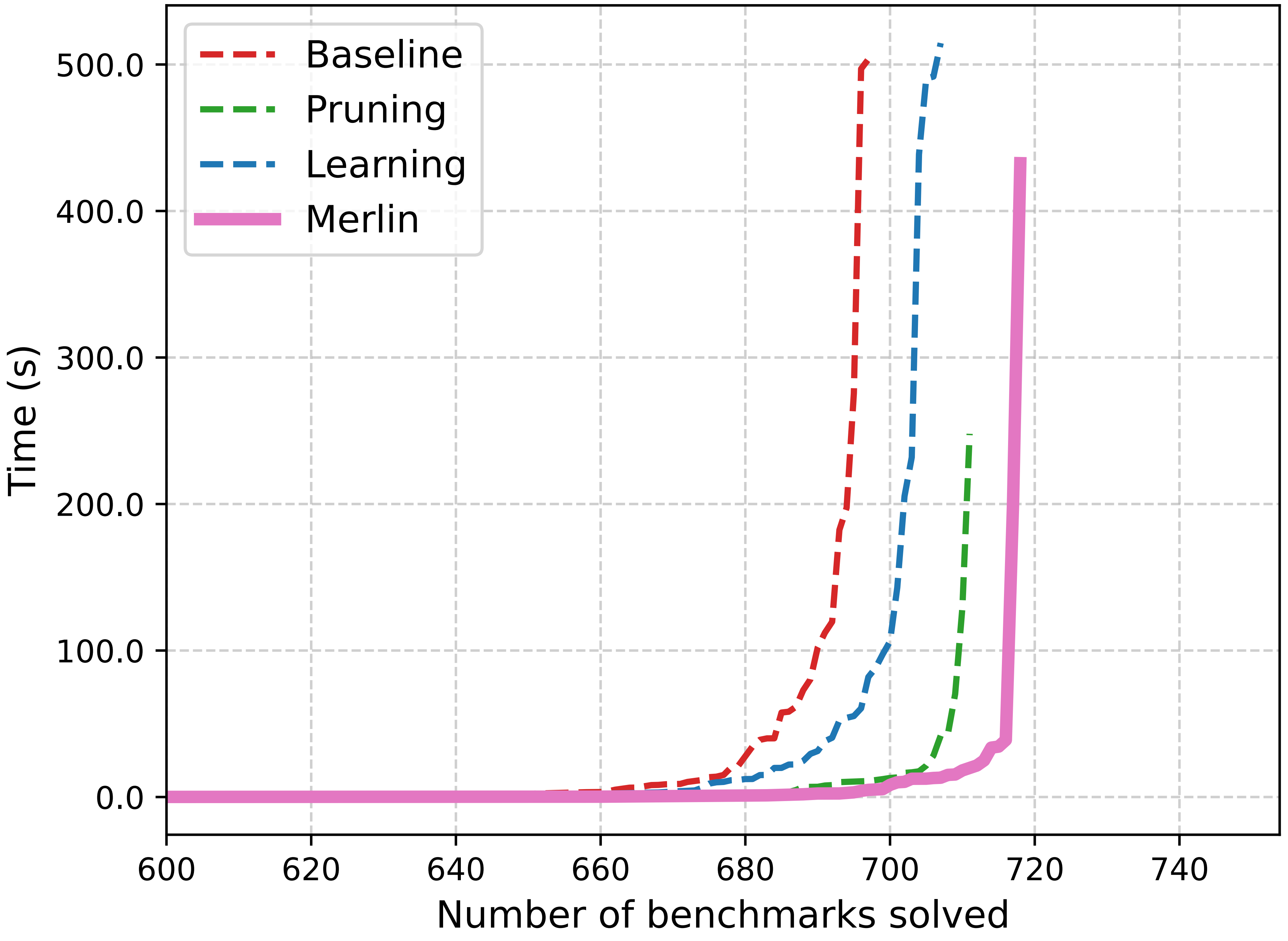
Blaze (String)

108 Blaze [Wang 2018]



75x faster than Blaze

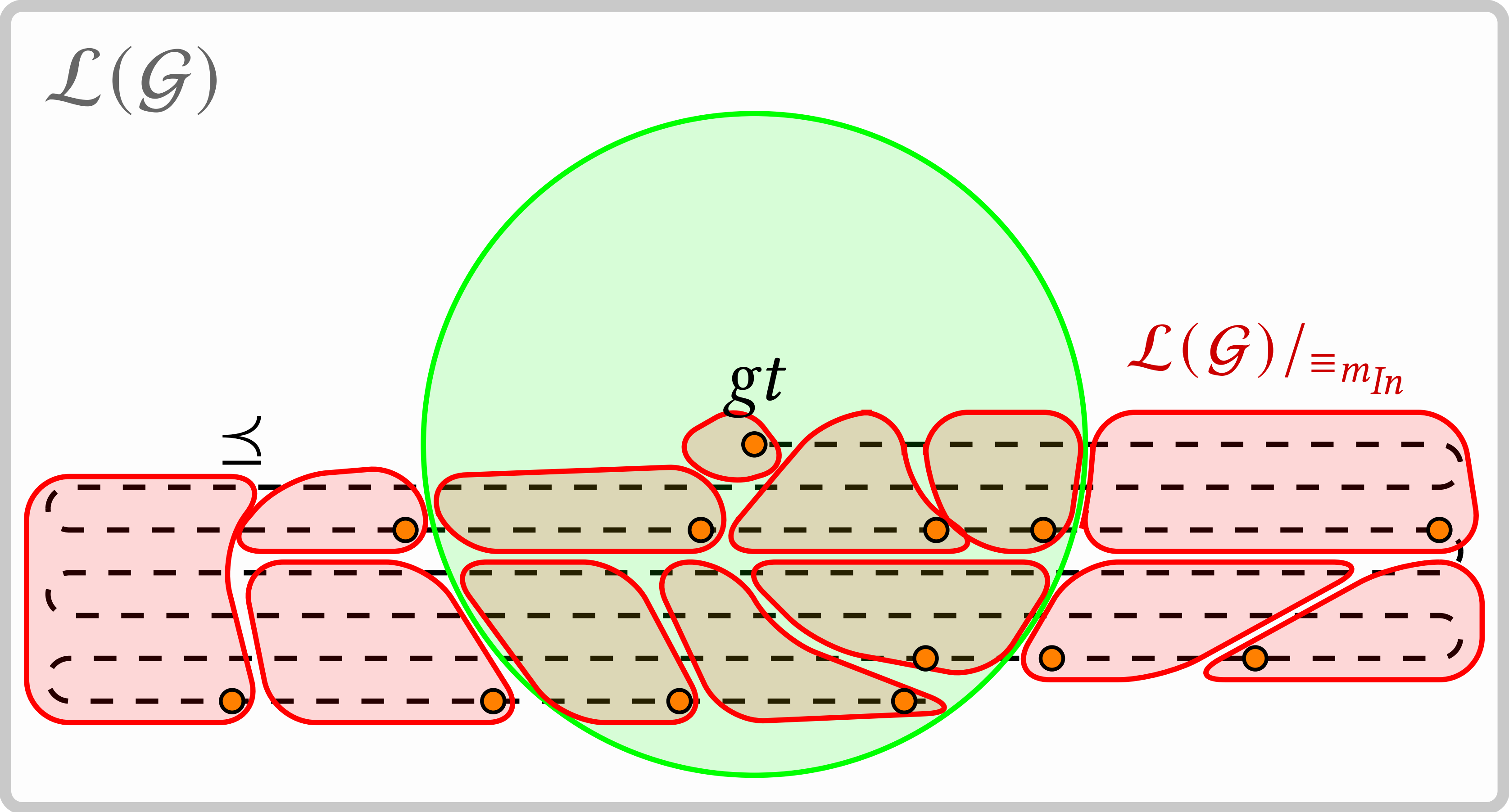
SyGuS-Ablation

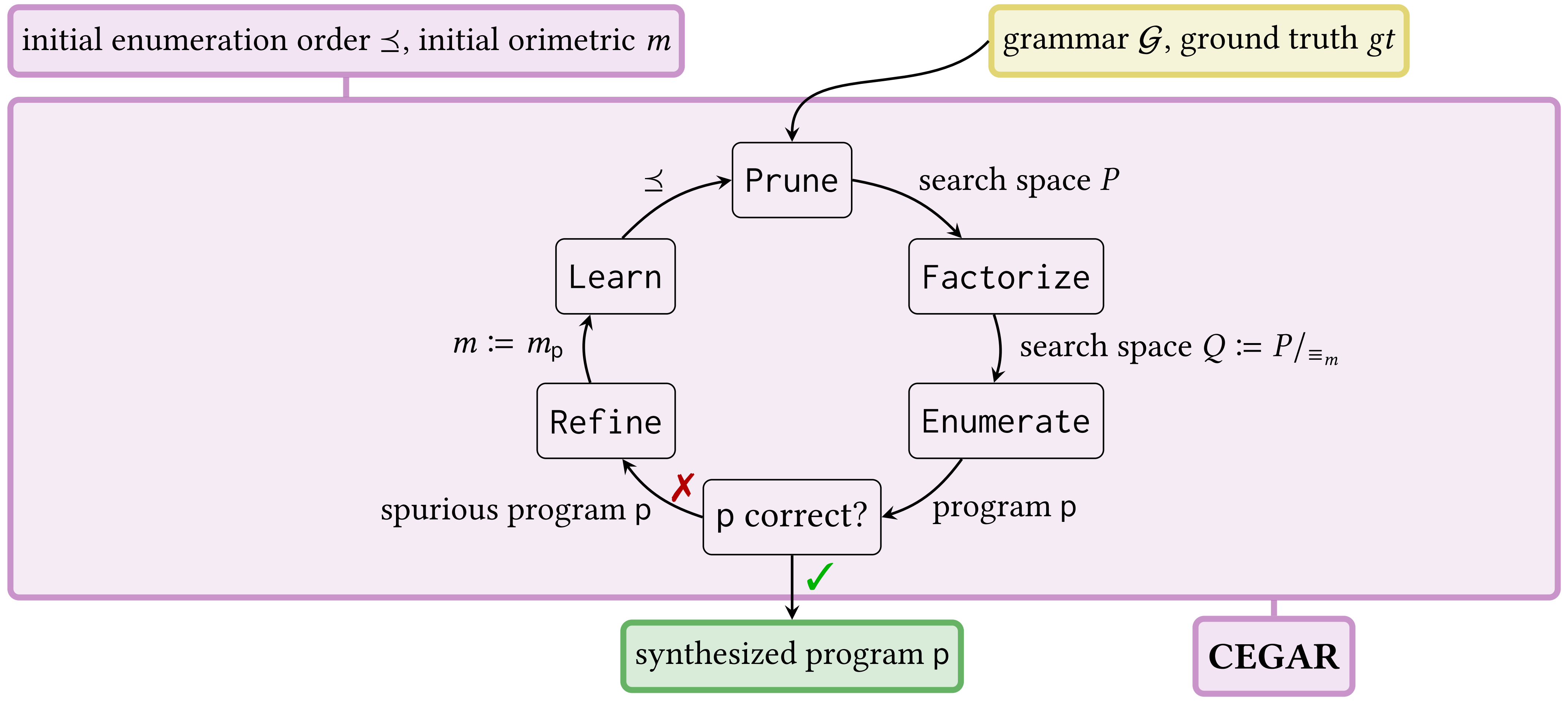


42x faster than Baseline

Conclusion

Framework for bottom-up enum. synthesis


$\mathcal{L}(\mathcal{G})$
gt
$\mathcal{L}(\mathcal{G})/_{\equiv_{m_{In}}}$
$\preceq$


initial enumeration order $\preceq$, initial orimetric m

grammar $\mathcal{G}$, ground truth gt

Prune

search space P

Factorize

search space $Q := P/_{\equiv_m}$

Enumerate

program p

p correct?

✓

synthesized program p

✗

spurious program p

Refine

$m := m_{\mathrm{p}}$

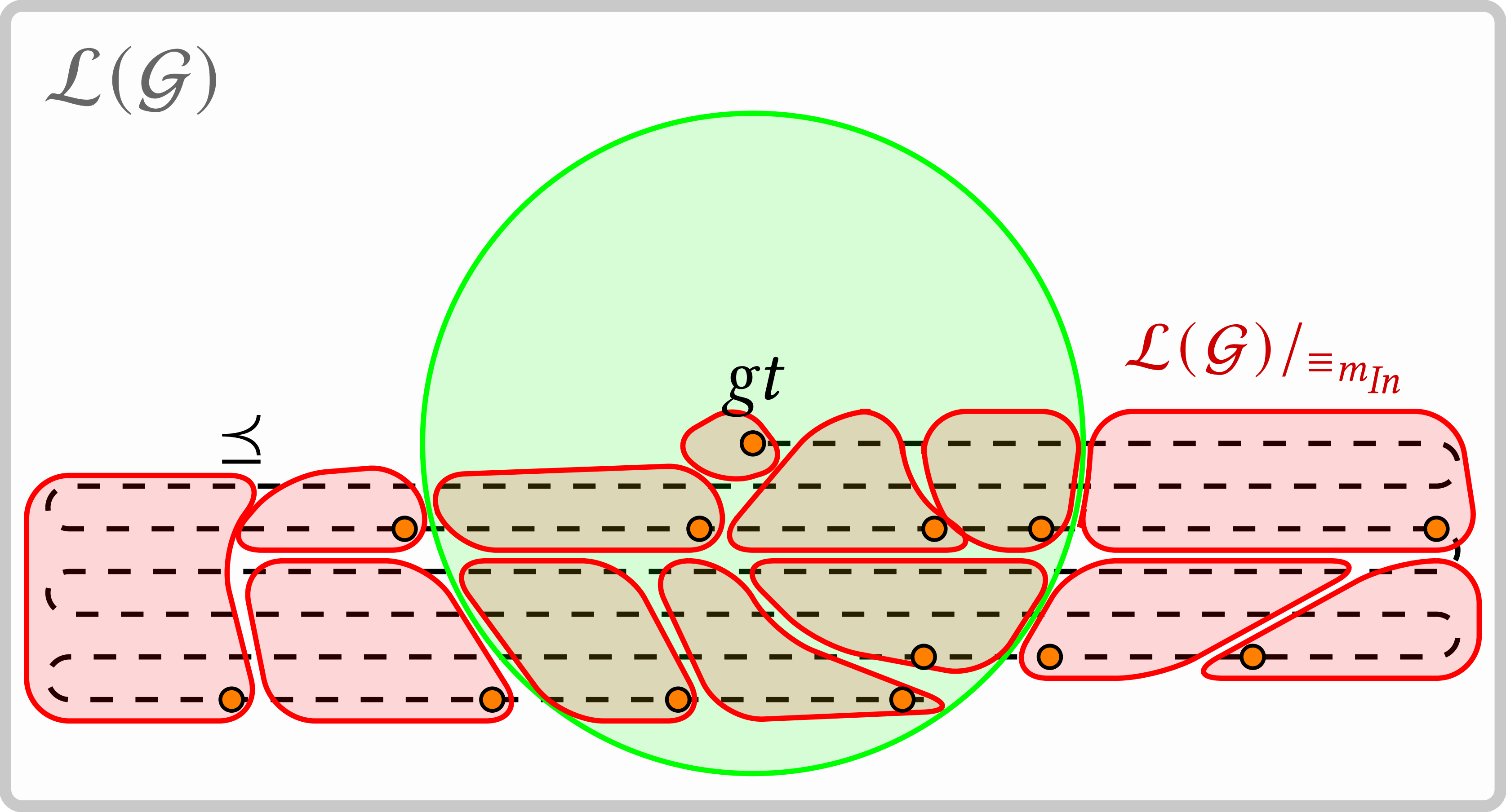
Learn

$\preceq$

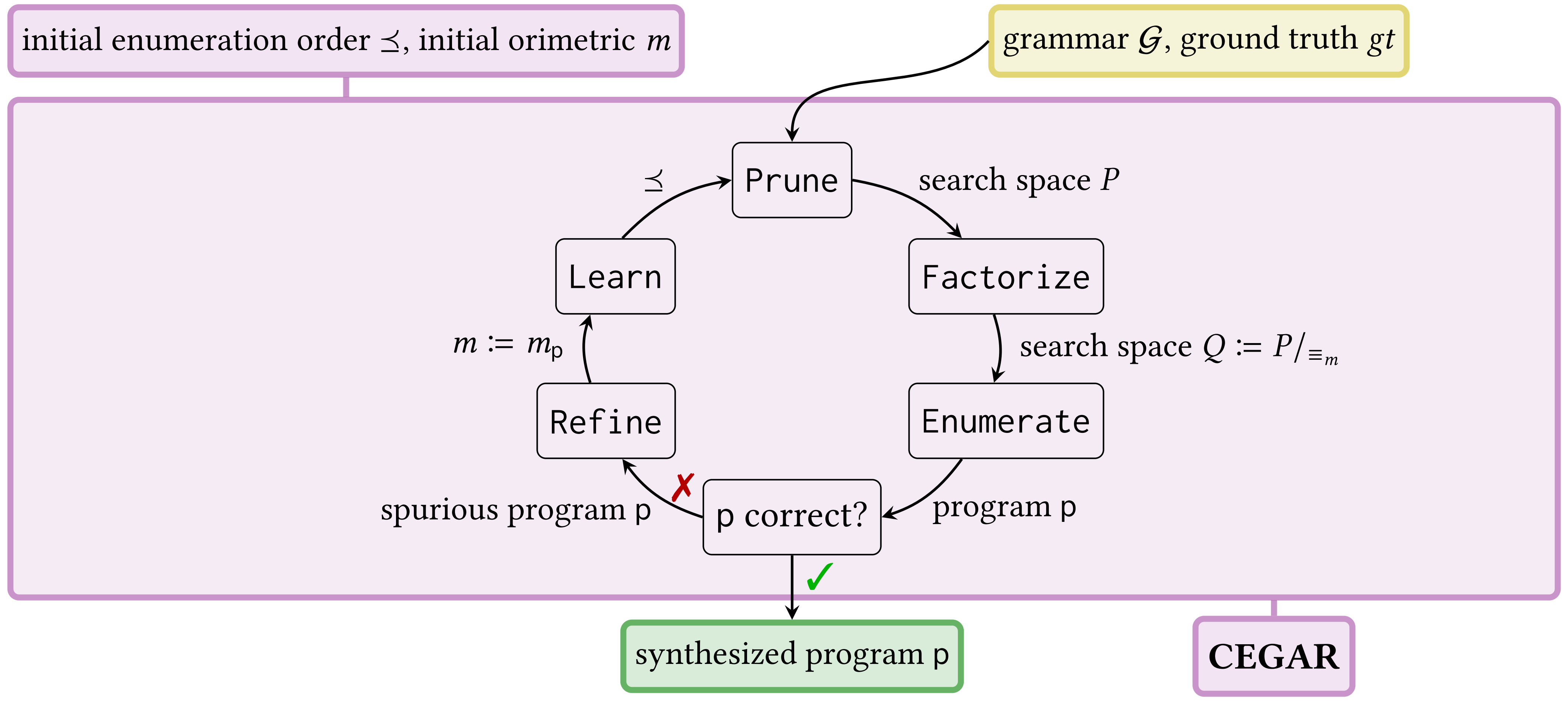
CEGAR

Conclusion

Framework for bottom-up enum. synthesis



$\mathcal{L}(\mathcal{G})$
gt
$\preceq$
$\mathcal{L}(\mathcal{G})/_{\equiv_{m_{In}}}$

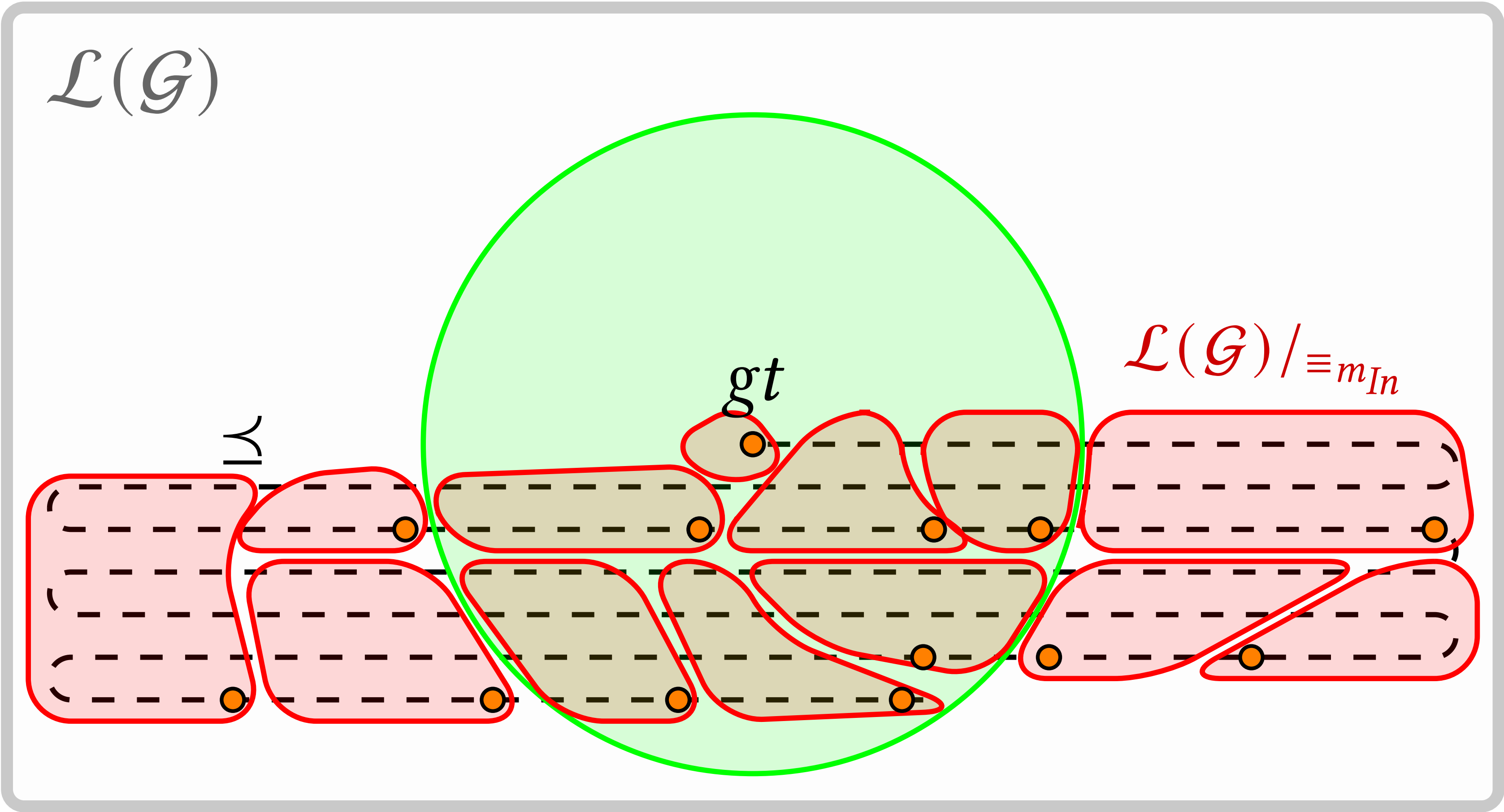


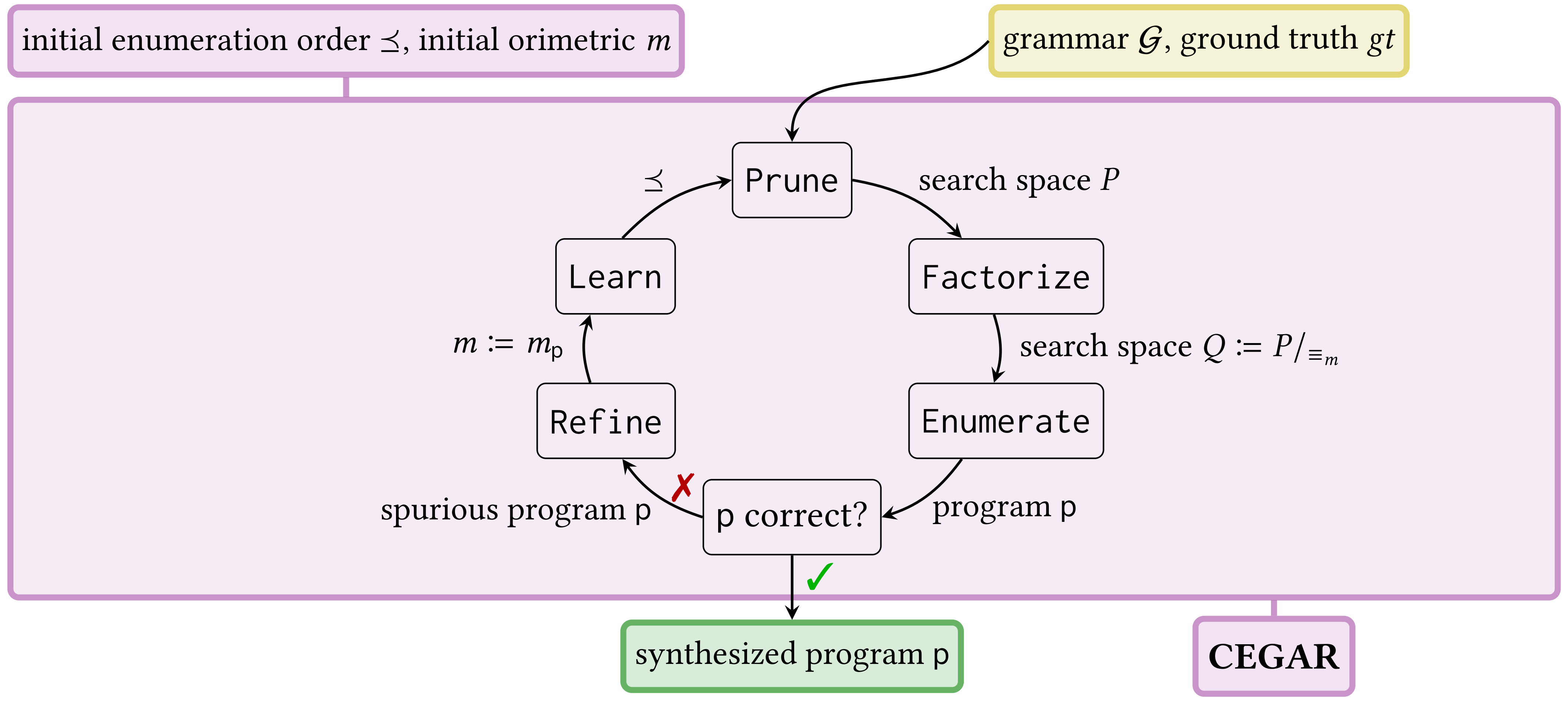
Oriented Metrics

$$\begin{aligned} && m(a,a) &= 0 && \text{(reflexivity)} \\ m(b,a) = 0 &\implies & m(a,b) &= 0 && \text{(symmetry at zero)} \\ & m(a,c) \leq & m(a,b) &+ m(b,c) && \text{($\triangle$-inequality)} \end{aligned}$$

Conclusion

Framework for bottom-up enum. synthesis


$\mathcal{L}(\mathcal{G})$
gt
$\mathcal{L}(\mathcal{G})/_{\equiv_{m_{In}}}$
$\preceq$



initial enumeration order $\preceq$, initial orimetric m

grammar $\mathcal{G}$, ground truth gt

Prune

search space P

Factorize

search space $Q := P/_{\equiv_m}$

Enumerate

program p

p correct?

✓ synthesized program p

✗ spurious program p

Refine

$m := m_p$

Learn

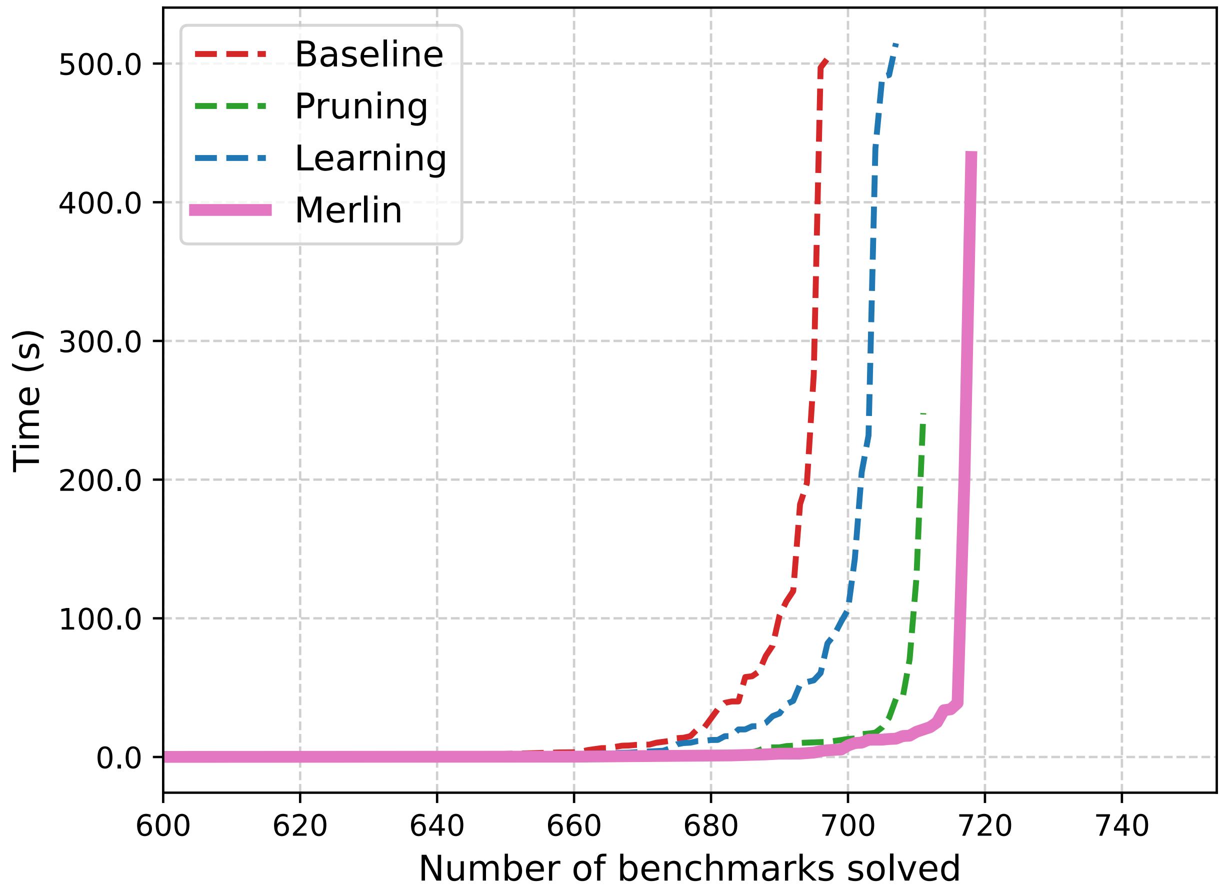
$\preceq$

CEGAR

Oriented Metrics

$$\begin{aligned} && m(a,a) &= 0 && \text{(reflexivity)} \\ m(b,a) = 0 &\;\Rightarrow & m(a,b) &= 0 && \text{(symmetry at zero)} \\ && m(a,c) &\leq m(a,b) + m(b,c) && \text{($\triangle$-inequality)} \end{aligned}$$

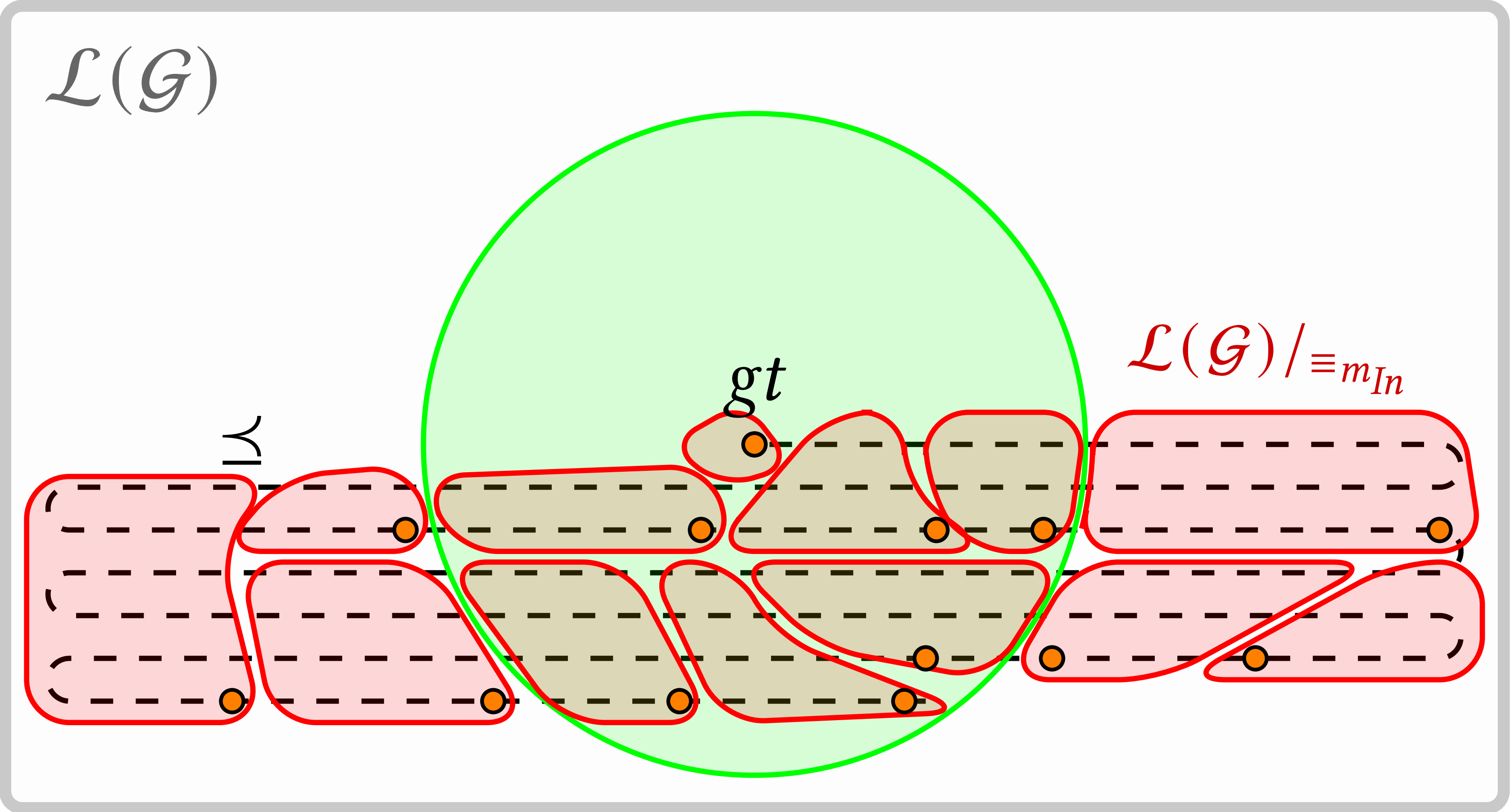
Substantial impact on performance

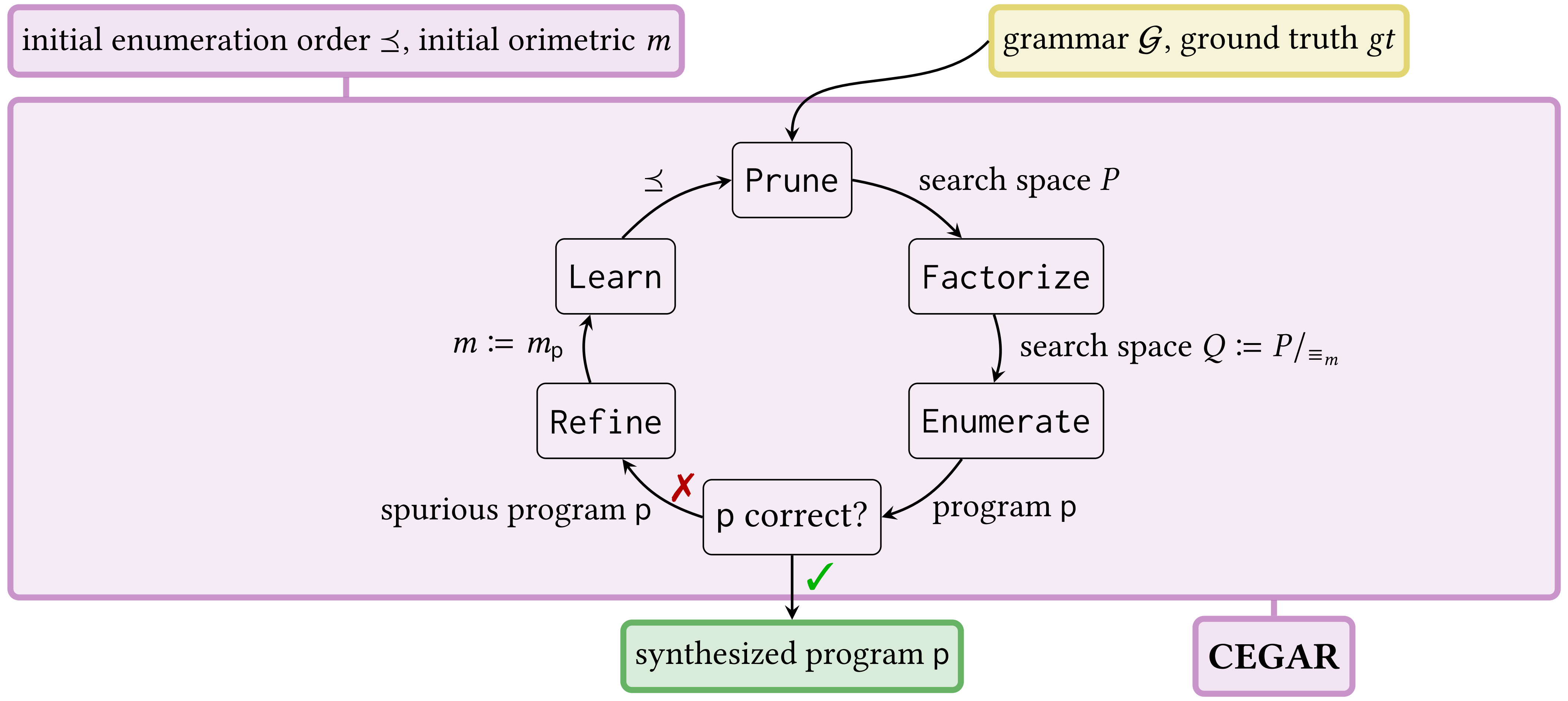


Conclusion

Thank you! Questions?

Framework for bottom-up enum. synthesis


$\mathcal{L}(\mathcal{G})$
gt
$\mathcal{L}(\mathcal{G})/_{\equiv_{m_{In}}}$
$\preceq$



initial enumeration order $\preceq$, initial orimetric m

grammar $\mathcal{G}$, ground truth gt

Prune

search space P

Factorize

search space $Q := P/_{\equiv_m}$

Enumerate

program p

p correct?

✓

synthesized program p

✗

spurious program p

Refine

$m := m_{\mathrm{p}}$

Learn

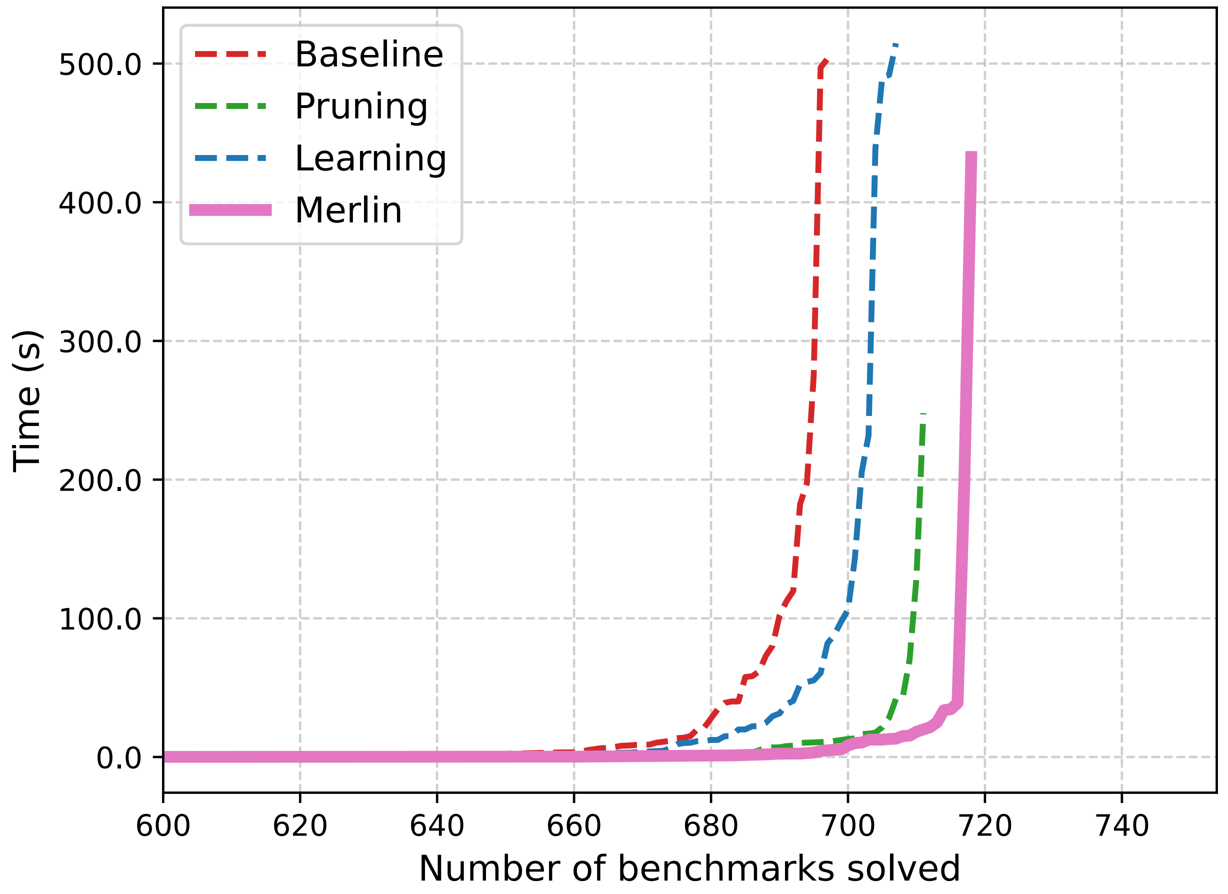
$\preceq$

CEGAR

Oriented Metrics

$$\begin{aligned} & & m(a,a) &= 0 & &\text{(reflexivity)}\\ m(b,a) = 0 &\Rightarrow & m(a,b) &= 0 & &\text{(symmetry at zero)}\\ m(a,c) &\leq & m(a,b) &+ m(b,c) & &\text{($\triangle$-inequality)} \end{aligned}$$

Substantial impact on performance


Baseline
Pruning
Learning
Merlin
Time (s)
Number of benchmarks solved