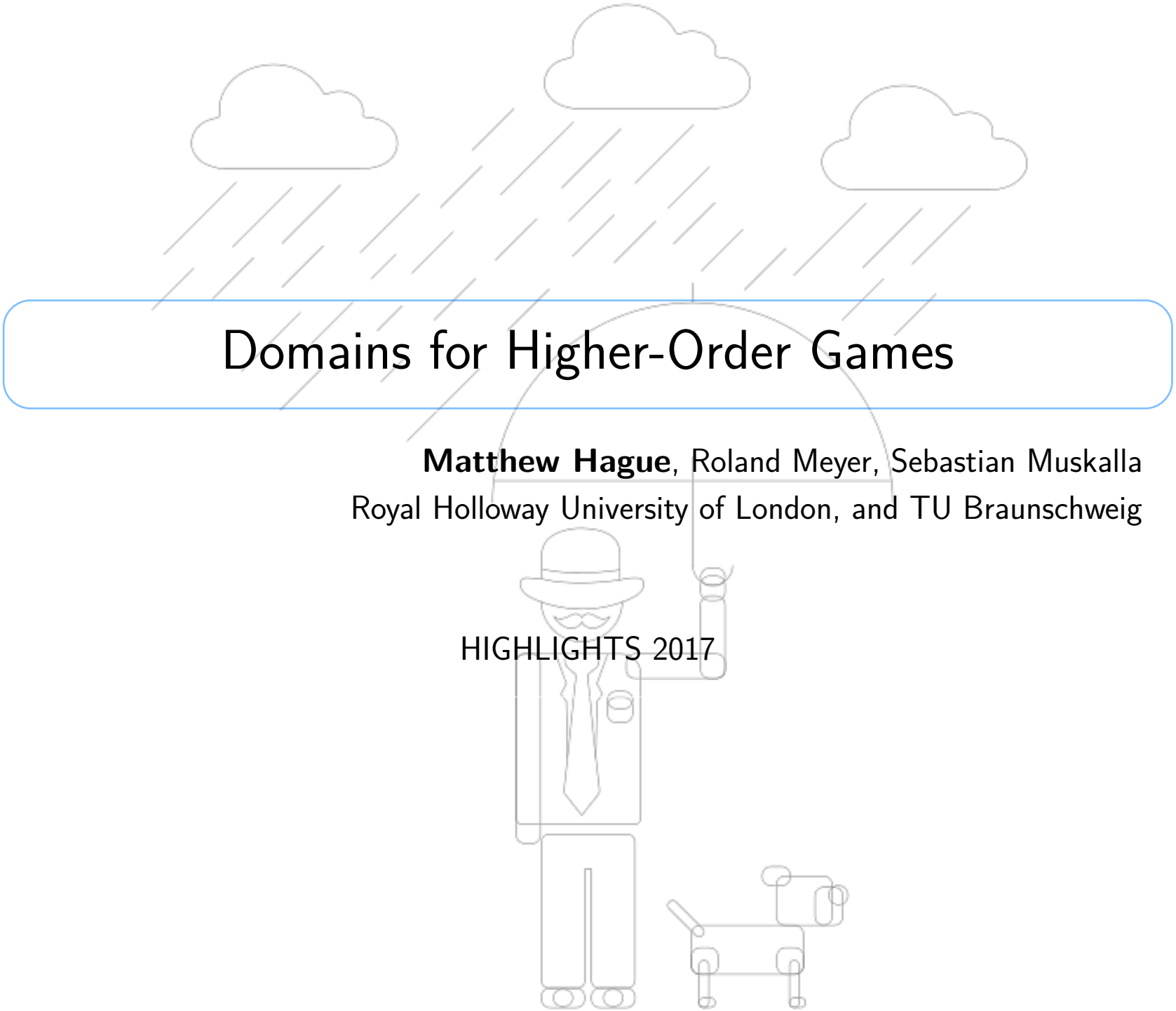


Domains for Higher-Order Games

Matthew Hague, Roland Meyer, Sebastian Muskalla
Royal Holloway University of London, and TU Braunschweig

HIGHLIGHTS 2017

Overview – The Problem

Decide the winning region / strategy of inclusion games

- Played over higher-order recursion schemes.
  - (Higher-order control-flow)

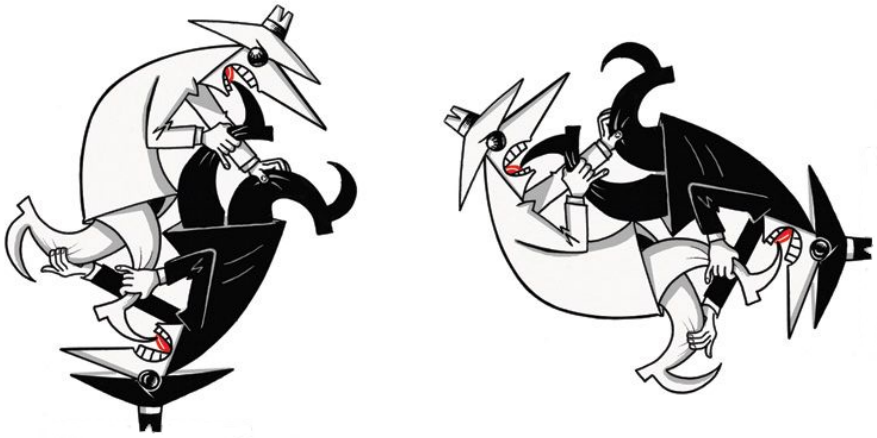
Overview – The Problem

Decide the winning region / strategy of inclusion games

- Played over higher-order recursion schemes.
  - (Higher-order control-flow)
- A play generates a program trace.

Overview – The Problem

Decide the winning region / strategy of inclusion games



- Played over higher-order recursion schemes.
  - (Higher-order control-flow)
- A play generates a program trace.

Overview – The Problem

Decide the winning region / strategy of inclusion games

- Played over higher-order recursion schemes.
  - (Higher-order control-flow)
- A play generates a program trace.

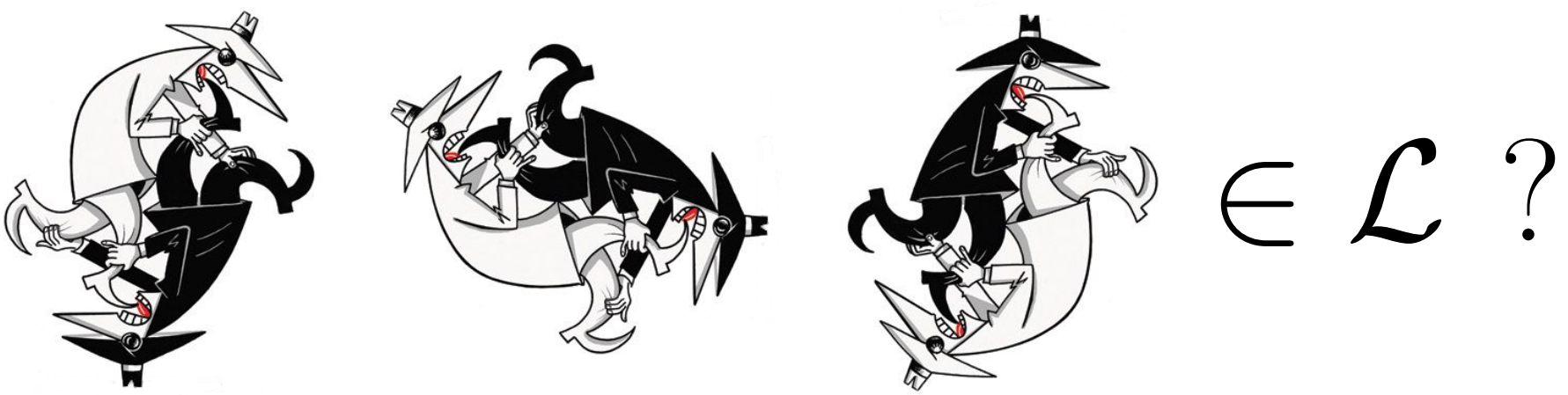
Overview – The Problem

Decide the winning region / strategy of inclusion games

- Played over higher-order recursion schemes.
  - (Higher-order control-flow)
- A play generates a program trace.
- The program trace must belong to a regular specification.

Overview – The Problem

Decide the winning region / strategy of inclusion games



- Played over higher-order recursion schemes.
  - (Higher-order control-flow)
- A play generates a program trace.
- The program trace must belong to a regular specification.

Overview – Our Solution

- **Semantics:** concrete $[\![S]\!]$ of G wrt $\mathcal{A}$
- **Abstract:** finite abstract semantics
- **Kleene iteration:** compute abstract semantics

Synthesis

Why write a bad program and check it's ok?

- Better to generate a correct program!

Synthesis

Why write a bad program and check it's ok?

- Better to generate a correct program!

The synthesis problem

Given:

- Template of a program P and a specification φ

Question:

- Is there an instantiation P' that satisfies φ?

Higher-Order Games: Input

Template program:

- Higher-Order Recursion Scheme
- Ownership partition of non-terminals

$$
\begin{aligned}
S_{\circ} &= F_{\circ}\ G_{\square} \\
F_{\circ}\ f &= a\ (F_{\circ}\ f) \vee a\ (f\ b) \\
G_{\square}\ x &= x \wedge b\ x
\end{aligned}
$$

Higher-Order Games: Input

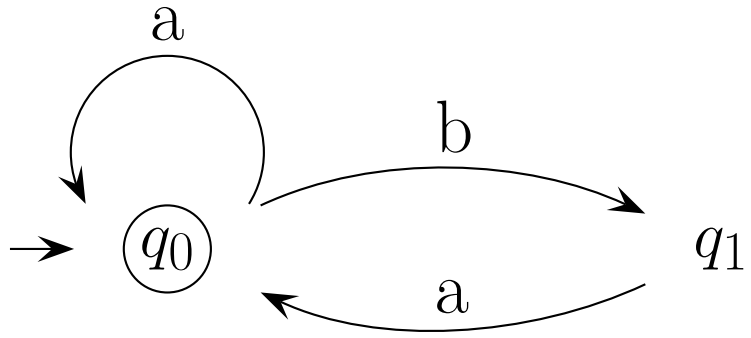
Template program:

- Higher-Order Recursion Scheme
- Ownership partition of non-terminals

$$\begin{aligned} S_\circ &= F_\circ\ G_\square \\ F_\circ\ f &= a\ (F_\circ\ f) \vee a\ (f\ b) \\ G_\square\ x &= x \wedge b\ x \end{aligned}$$

Specification (allowed traces):

- Finite automaton $\mathcal{A}$ over terminals ($\{a, b\}$)



Safety Games

We study safety games:

Can Player $\circ$ avoid generating a word $w \notin \mathcal{L}_{\mathcal{A}}$?

Example Play

$$
\begin{aligned}
S_{\circ} &= F_{\circ}\ G_{\square} \\
F_{\circ}\ f &= a\ (F_{\circ}\ f) \vee a\ (f\ b) \qquad\qquad S_{\circ} \\
G_{\square}\ x &= x \wedge b\ x
\end{aligned}
$$

Example Play

$$S_\circ = F_\circ\ G_\square$$

$$F_\circ\ f = a\ (F_\circ\ f) \vee a\ (f\ b) \qquad S_\circ$$

$$G_\square\ x = x \wedge b\ x$$

Example Play

$$S_\circ = F_\circ\ G_\square$$

$$F_\circ\ f = a\ (F_\circ\ f) \vee a\ (f\ b)$$

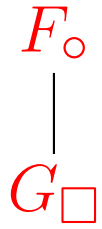
$$G_\square\ x = x \wedge b\ x$$

Example Play

$$S_{\circ} = F_{\circ}\ G_{\square}$$

$$F_{\circ}\ f = a\ (F_{\circ}\ f) \vee a\ (f\ b)$$

$$G_{\square}\ x = x \wedge b\ x$$

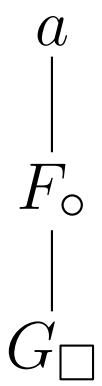


Example Play

$$S_\circ = F_\circ\ G_\square$$

$$F_\circ\ f = a\ (F_\circ\ f) \vee a\ (f\ b)$$

$$G_\square\ x = x \wedge b\ x$$

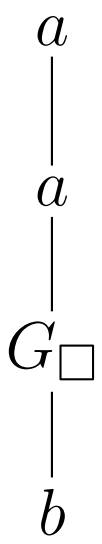


Example Play

$$
\begin{aligned}
S_{\circ} &= F_{\circ}\ G_{\square} \\
F_{\circ}\ f &= a\ (F_{\circ}\ f) \vee a\ (f\ b) \\
G_{\square}\ x &= x \wedge b\ x
\end{aligned}
$$

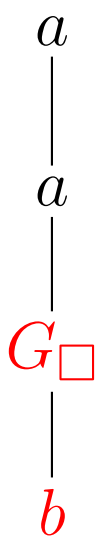
Example Play

$$
\begin{aligned}
S_{\circ} &= F_{\circ}\ G_{\square} \\
F_{\circ}\ f &= a\ (F_{\circ}\ f) \vee a\ (f\ b) \\
G_{\square}\ x &= x \wedge b\ x
\end{aligned}
$$



Example Play

$$S_\circ = F_\circ\ G_\square$$
$$F_\circ\ f = a\ (F_\circ\ f) \vee a\ (f\ b)$$
$$G_\square\ x = x \wedge b\ x$$



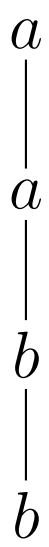
Example Play

$$
\begin{aligned}
S_{\circ} &= F_{\circ}\ G_{\Box} \\
F_{\circ}\ f &= a\ (F_{\circ}\ f) \vee a\ (f\ b) \\
G_{\Box}\ x &= x \wedge b\ x
\end{aligned}
$$

a
|
a
|
b
|
b

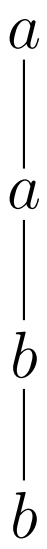
Example Play

$$
\begin{aligned}
S_{\circ} &= F_{\circ}\ G_{\square} \\
F_{\circ}\ f &= a\ (F_{\circ}\ f) \vee a\ (f\ b) \\
G_{\square}\ x &= x \wedge b\ x
\end{aligned}
$$



Example Play

$$
\begin{aligned}
S_{\circ} &= F_{\circ}\ G_{\square} \\
F_{\circ}\ f &= a\ (F_{\circ}\ f) \vee a\ (f\ b) \\
G_{\square}\ x &= x \wedge b\ x
\end{aligned}
$$



Since $aabb \notin \overline{\Sigma^* bb \Sigma^*}$ Player $\circ$ loses this play.

Results

Theorem

Given

- order-k game G
- regular specification $\mathcal{A}$

deciding G wrt $\mathcal{A}$ is

$$(k+1)\text{-EXPTIME-complete}$$

Results

Theorem

Given

- order-k game G
- regular specification $\mathcal{A}$

deciding G wrt $\mathcal{A}$ is

$$(k+1)\text{-EXPTIME-complete}$$

Such a result is already known

- Determinize $\mathcal{A}$, product with G
- $\Rightarrow$ standard safety game over higher-order recursion schemes.
  - Solvable by e.g. [Serre]

Our Approach

We take a different approach

- **Semantics:** concrete $[\![S]\!]$ of G wrt $\mathcal{A}$
- **Abstract:** finite abstract semantics
- **Kleene iteration:** compute abstract semantics

Our Approach

We take a different approach

- **Semantics:** concrete $[\![S]\!]$ of G wrt $\mathcal{A}$
  - infinite CPPO: monotone boolean formulas and continuous (higher-order) functions between them.
- **Abstract:** finite abstract semantics

- **Kleene iteration:** compute abstract semantics

Our Approach

We take a different approach

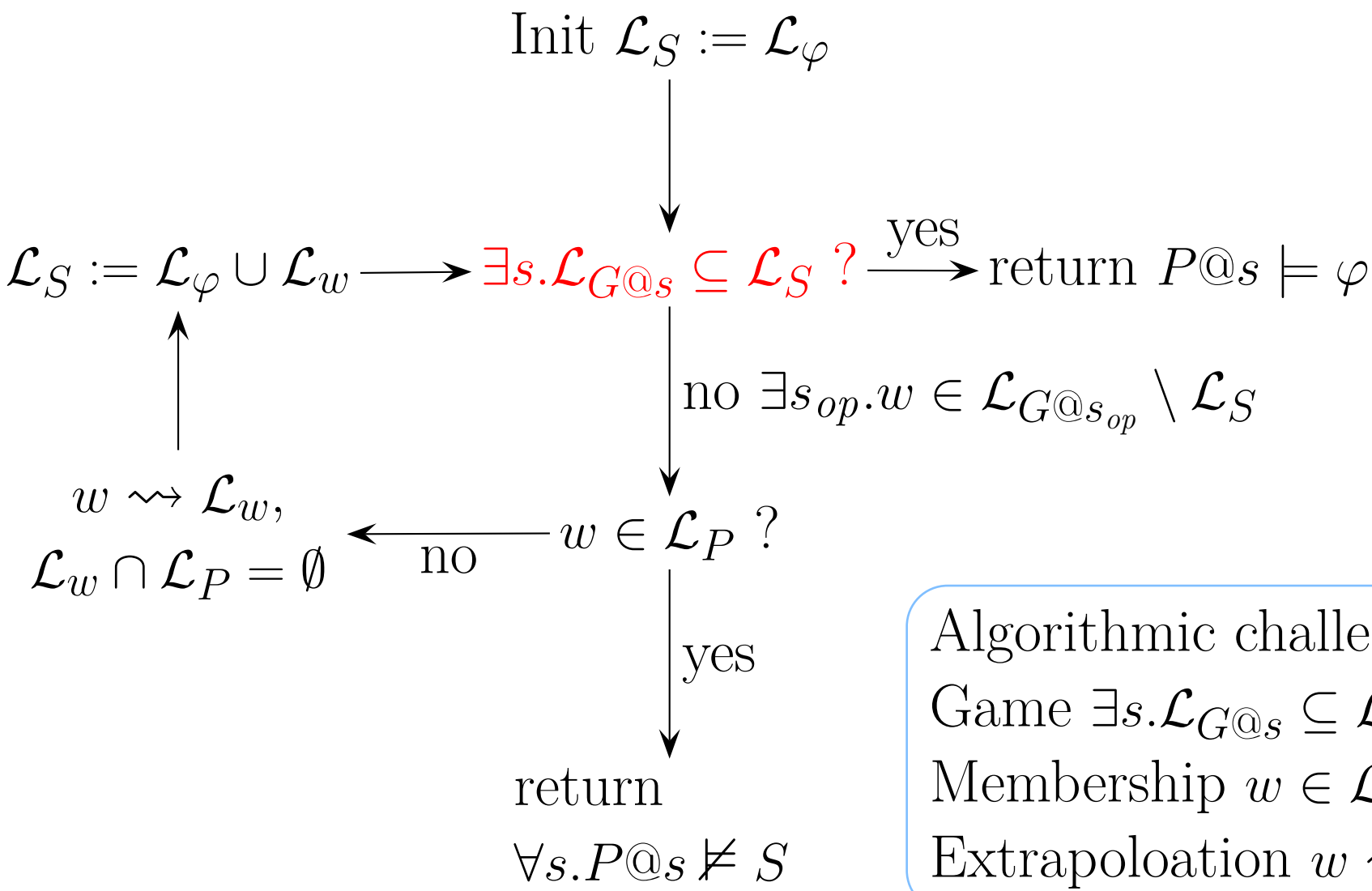
- **Semantics:** concrete $[\![S]\!]$ of G wrt $\mathcal{A}$
  - infinite CPPO: monotone boolean formulas and continuous (higher-order) functions between them.
- **Abstract:** finite abstract semantics
  - Framework for exact abstract interpretation
- **Kleene iteration:** compute abstract semantics

Related Work

Similar approaches have been studied in the literature.

- Models/domains:
  - Walukiewicz & Salvati
  - Melliès & Grellois
  - Hofmann, Chen & Ledent
- Abstract interpretation:
  - Abramsky & Hankin
  - Ramsay
  - Hofmann, Chen & Ledent

Podelski's CEGAR Loop



Algorithmic challenges:

Game $\exists s.\mathcal{L}_{G@s} \subseteq \mathcal{L}_S$

Membership $w \in \mathcal{L}_P$

Extrapoloation $w \rightsquigarrow \mathcal{L}_w$

Conclusion

We have

- Defined and motivated higher-order inclusion games
- Shown $(k+1)$-EXPTIME-completeness
- Given a solution based on semantics in CPPOs
- Used exact abstract interpretation to obtain an effective (and optimised) solution

Future work

- Categories?
- More powerful winning conditions