# Characteristic Formulae for Session Types

**Julien Lange** Nobuko Yoshida

Imperial College London

BETTY Meeting - March 2016

# Motivations

Reverse-engineering and **modernisation of legacy systems** with session types, e.g., from COBOL programs to cloud-based solutions.

# Introduction

**Session types**: a theory to reason about concurrent programs
. . . applied to reverse-engineering $\Rightarrow$ needs to scale

A crucial ingredient of the theory is the notion of **subtyping**

- Inevitable in programming languages
- Relates to simulations and pre-orders in automata and process algebra theories.
- **Characteristic formulae** approach*: a link between session type theory and model checking.

$$\underbrace{P \preceq Q}_{\text{simulation relation}} \quad \Longleftrightarrow \quad \underbrace{Q \models \mathcal{X}(P)}_{\text{model checking}}$$

*Bernhard Steffen. *Characteristic formulae* (ICALP 1989)

# Plan

Introduction

Session types & subtyping

Modal $\mu$-calculus

Characteristic formulae for subtyping

Evaluation

Conclusions

# Session types in one slide

```
// Client-Side
int x; boolean y;
Session s = MyAPI.Session("my-public-port");
if (...):  // e.g., user input
   s.send("d"); s.send(3); s.receive(x);
else:
   s.send("n"); s.send(true); s.receive(y);
```

```
// Server-Side
int x; boolean y;
Session s = MyAPI.Session("my-public-port");
switch (s.receive(x)):
   case "d": s.receive(x); s.send(x*2);
   case "n": s.receive(y); s.send(not(y));
```

# Session types in one slide

```
// Client-Side
int x; boolean y;
Session s = MyAPI.Session("my-public-port");
if (...):  // e.g., user input
   s.send("d"); s.send(3); s.receive(x);
else:
   s.send("n"); s.send(true); s.receive(y);
```

$$T = (!d.!int.?int.\mathtt{end}) \oplus (!n.!bool.?bool.\mathtt{end})$$

```
// Server-Side
int x; boolean y;
Session s = MyAPI.Session("my-public-port");
switch (s.receive(x)):
   case "d": s.receive(x); s.send(x*2);
   case "n": s.receive(y); s.send(not(y));
```

# Session types in one slide

```
// Client-Side
int x; boolean y;
Session s = MyAPI.Session("my-public-port");
if (...):  // e.g., user input
   s.send("d"); s.send(3); s.receive(x);
else:
   s.send("n"); s.send(true); s.receive(y);
```

$$T = (!d.!int.?int.\mathtt{end}) \oplus (!n.!bool.?bool.\mathtt{end})$$

```
// Server-Side
int x; boolean y;
Session s = MyAPI.Session("my-public-port");
switch (s.receive(x)):
   case "d": s.receive(x); s.send(x*2);
   case "n": s.receive(y); s.send(not(y));
```

$$S = (?d.?int.!int.\mathtt{end}) \,\&\, (?n.?bool.!bool.\mathtt{end})$$

# Session types in one slide

```
// Client-Side
int x; boolean y;
Session s = MyAPI.Session("my-public-port");
if (...):  // e.g., user input
   s.send("d"); s.send(3); s.receive(x);
else:
   s.send("n"); s.send(true); s.receive(y);
```

$$T = (!d.!int.?int.\text{end}) \oplus (!n.!bool.?bool.\text{end})$$

```
// Server-Side
int x; boolean y;
Session s = MyAPI.Session("my-public-port");
switch (s.receive(x)):
   case "d": s.receive(x); s.send(x*2);
   case "n": s.receive(y); s.send(not(y));
```

$$S = (?d.?int.!int.\text{end}) \,\&\, (?n.?bool.!bool.\text{end})$$

Session types $T$ and $S$ are **dual** of each other $\Rightarrow$ **safety**.

# Subtyping for session types

Syntax:

$$T := \bigoplus_{i \in I} !a_i.T_i \mid \&_{i \in I} ?a_i.T_i \mid \texttt{rec x}.T \mid \texttt{x} \mid \texttt{end}$$

# Subtyping for session types

Syntax:

$$T := \bigoplus_{i \in I} !a_i.T_i \mid \&_{i \in I} ?a_i.T_i \mid \mathtt{rec\,x}.T \mid \mathtt{x} \mid \mathtt{end}$$

Example of subtyping $\leq$:

$$\underbrace{!d.!int.?int.\mathtt{end}}_{T_1} \leq \underbrace{(!d.!int.?int.\mathtt{end}) \oplus (!n.!bool.?bool.\mathtt{end})}_{T_2}$$

$$\underbrace{(?d.?int.!int.\mathtt{end}) \& (?n.?bool.!bool.\mathtt{end})}_{S_1} \leq \underbrace{?n.?bool.!bool.\mathtt{end}}_{S_2}$$

# Subtyping for session types

Syntax:

$$T := \bigoplus_{i \in I} !a_i.T_i \mid \&_{i \in I} ?a_i.T_i \mid \mathtt{rec\,x}.T \mid \mathtt{x} \mid \mathtt{end}$$

Example of subtyping $\leq$:

$$\underbrace{!d.!int.?int.\mathtt{end}}_{T_1} \leq \underbrace{(!d.!int.?int.\mathtt{end}) \oplus (!n.!bool.?bool.\mathtt{end})}_{T_2}$$

$$\underbrace{(?d.?int.!int.\mathtt{end}) \& (?n.?bool.!bool.\mathtt{end})}_{S_1} \leq \underbrace{?n.?bool.!bool.\mathtt{end}}_{S_2}$$

If $S \mid T$ is safe and $U \leq S$, then $U \mid T$ is also safe.

# Subtyping for session types

Syntax:

$$T := \bigoplus_{i \in I} !a_i.T_i \mid \&_{i \in I} ?a_i.T_i \mid \mathtt{rec\,x}.T \mid \mathtt{x} \mid \mathtt{end}$$

Example of subtyping $\leq$:

$$\underbrace{!d.!int.?int.\mathtt{end}}_{T_1} \leq \underbrace{(!d.!int.?int.\mathtt{end}) \oplus (!n.!bool.?bool.\mathtt{end})}_{T_2}$$

$$\underbrace{(?d.?int.!int.\mathtt{end}) \& (?n.?bool.!bool.\mathtt{end})}_{S_1} \leq \underbrace{?n.?bool.!bool.\mathtt{end}}_{S_2}$$

If $S \mid T$ is safe and $U \leq S$, then $U \mid T$ is also safe.

## Example

$T_2 \mid S_1$ is safe (dual of each other), hence $T_1 \mid S_1$ is also safe.

# Subtyping rules

The binary relation $\leq$ is the **largest** relation that contains the rules:

$$\frac{I \subseteq J \quad \forall i \in I \,:\, T_i \leq U_i}{\bigoplus_{i \in I} !a_i.T_i \;\leq\; \bigoplus_{j \in J} !a_j.U_j} \qquad \frac{J \subseteq I \quad \forall j \in J \,:\, T_j \leq U_j}{\&_{i \in I} ?a_i.T_i \;\leq\; \&_{j \in J} ?a_j.U_j}$$

$$\frac{}{\texttt{end} \;\leq\; \texttt{end}}$$

*The rules are interpreted **co-inductively** with an **equi-recursive** view of session types.*

# Subtyping rules

The binary relation $\leq$ is the **largest** relation that contains the rules:

$$\frac{I \subseteq J \quad \forall i \in I \,:\, T_i \leq U_i}{\bigoplus_{i \in I} !a_i.T_i \;\leq\; \bigoplus_{j \in J} !a_j.U_j} \qquad \frac{J \subseteq I \quad \forall j \in J \,:\, T_j \leq U_j}{\&_{i \in I} ?a_i.T_i \;\leq\; \&_{j \in J} ?a_j.U_j}$$

$$\frac{}{\mathtt{end} \;\leq\; \mathtt{end}}$$

*The rules are interpreted **co-inductively** with an **equi-recursive** view of session types.*

## Example

$$\dfrac{\dfrac{\vdots}{!int.?int.\mathtt{end} \;\leq\; !int.?int.\mathtt{end}}}{!d.!int.?int.\mathtt{end} \;\leq\; (!d.!int.?int.\mathtt{end}) \oplus (!n.!bool.?bool.\mathtt{end})}$$

# Subtyping rules

The binary relation $\leq$ is the **largest** relation that contains the rules:

$$\frac{I \subseteq J \quad \forall i \in I \,:\, T_i \leq U_i}{\bigoplus_{i \in I} !a_i.T_i \;\leq\; \bigoplus_{j \in J} !a_j.U_j} \qquad \frac{J \subseteq I \quad \forall j \in J \,:\, T_j \leq U_j}{\&_{i \in I} ?a_i.T_i \;\leq\; \&_{j \in J} ?a_j.U_j}$$

$$\frac{}{\mathtt{end} \;\leq\; \mathtt{end}}$$

*The rules are interpreted **co-inductively** with an **equi-recursive** view of session types.*

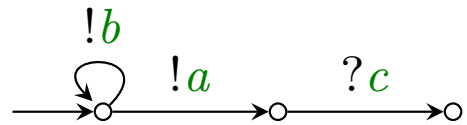
## Example

$$\dfrac{\dfrac{\vdots}{?bool.!bool.\mathtt{end} \;\leq\; ?bool.!bool.\mathtt{end}}}{(?d.?int.!int.\mathtt{end}) \,\&\, (?n.?bool.!bool.\mathtt{end}) \;\leq\; ?n.?bool.!bool.\mathtt{end}}$$

# Modal $\mu$-calculus

A session type $T$ induces a **finite** LTS, e.g.,

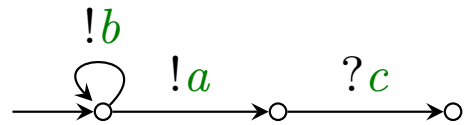
$\mathtt{rec}\,\mathtt{x}.(!a.?c.\mathtt{end}) \oplus (!b.\mathtt{x})$ gives



# Modal $\mu$-calculus

A session type $T$ induces a **finite** LTS, e.g.,

$\texttt{rec}\,\mathbf{x}.(!a.\,?c.\texttt{end})\oplus(!b.\mathbf{x})$ gives



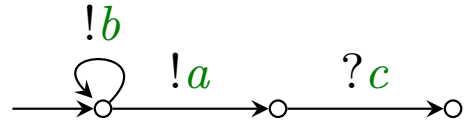

We can interpret a $\mu$-calculus formula $\phi$ on a type $T$, with $T \models \phi$.

$$\phi \;:=\; \top \mid \bot \mid \phi \wedge \phi \mid \phi \vee \phi \mid$$
$$[\dagger a]\phi \mid \langle \dagger a \rangle \phi \mid \nu \mathbf{x}.\,\phi \mid \mathbf{x} \qquad \text{with } \dagger a := !a \mid ?a$$

# Modal $\mu$-calculus

A session type $T$ induces a **finite** LTS, e.g.,

$\mathtt{rec}\,\mathbf{x}.(!a.?c.\mathtt{end})\oplus(!b.\mathbf{x})$ gives

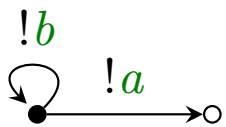



We can interpret a $\mu$-calculus formula $\phi$ on a type $T$, with $T \models \phi$.

$$\phi \;:=\; \top \mid \bot \mid \phi \wedge \phi \mid \phi \vee \phi \mid$$
$$[\dagger a]\phi \mid \langle \dagger a \rangle \phi \mid \nu \mathbf{x}.\,\phi \mid \mathbf{x} \qquad \text{with } \dagger a := !a \mid ?a$$

## Example

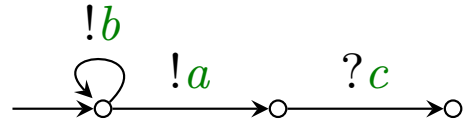
$\mathtt{rec}\,\mathbf{x}.(!a.\mathtt{end})\oplus(!b.\mathbf{x}) \models$



# Modal $\mu$-calculus

A session type $T$ induces a **finite** LTS, e.g.,

$\mathtt{rec}\,\mathbf{x}.(!a.?c.\mathtt{end})\oplus(!b.\mathbf{x})$ gives

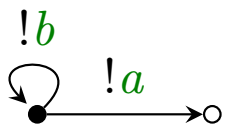



We can interpret a $\mu$-calculus formula $\phi$ on a type $T$, with $T \models \phi$.

$$\phi \;:=\; \top \mid \bot \mid \phi \wedge \phi \mid \phi \vee \phi \mid$$
$$[\dagger a]\phi \mid \langle \dagger a \rangle \phi \mid \nu \mathbf{x}.\,\phi \mid \mathbf{x} \qquad \text{with } \dagger a := !a \mid ?a$$

## Example

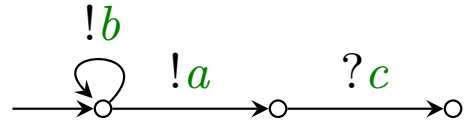
$\mathtt{rec}\,\mathbf{x}.(!a.\mathtt{end})\oplus(!b.\mathbf{x}) \;\models\; \langle !a \rangle \top \wedge \langle !d \rangle \top$ ✗



# Modal $\mu$-calculus

A session type $T$ induces a **finite** LTS, e.g.,

$\mathtt{rec}\,\mathbf{x}.(!a.?c.\mathtt{end})\oplus(!b.\mathbf{x})$ gives

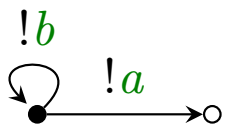



We can interpret a $\mu$-calculus formula $\phi$ on a type $T$, with $T \models \phi$.

$$\phi \;:=\; \top \mid \bot \mid \phi \wedge \phi \mid \phi \vee \phi \mid$$
$$[\dagger a]\phi \mid \langle \dagger a \rangle \phi \mid \nu \mathbf{x}.\,\phi \mid \mathbf{x} \qquad \text{with } \dagger a := !a \mid ?a$$

## Example

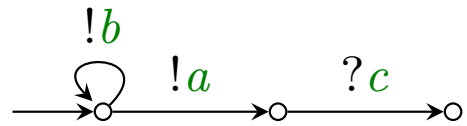
$\mathtt{rec}\,\mathbf{x}.(!a.\mathtt{end})\oplus(!b.\mathbf{x}) \;\models\; [!c]\top \vee [!d]\bot$ ✓



# Modal $\mu$-calculus

A session type $T$ induces a **finite** LTS, e.g.,

$\mathtt{rec}\,\mathbf{x}.(!a.?c.\mathtt{end})\oplus(!b.\mathbf{x})$ gives

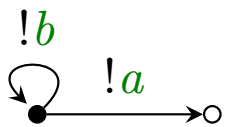



We can interpret a $\mu$-calculus formula $\phi$ on a type $T$, with $T \models \phi$.

$$\phi \;:=\; \top \mid \bot \mid \phi \wedge \phi \mid \phi \vee \phi \mid$$
$$[\dagger a]\phi \mid \langle \dagger a \rangle \phi \mid \nu \mathbf{x}.\,\phi \mid \mathbf{x} \qquad \text{with } \dagger a := !a \mid ?a$$

## Example

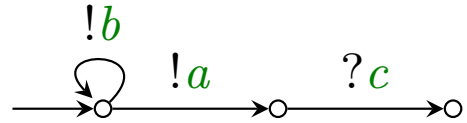
$\mathtt{rec}\,\mathbf{x}.(!a.\mathtt{end})\oplus(!b.\mathbf{x}) \;\models\; \nu \mathbf{x}.\,\langle !b \rangle \mathbf{x}$ ✓



# Modal $\mu$-calculus

A session type $T$ induces a **finite** LTS, e.g.,

$\mathtt{rec}\,\mathbf{x}.(!a.?c.\mathtt{end})\oplus(!b.\mathbf{x})$ gives

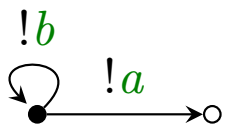



We can interpret a $\mu$-calculus formula $\phi$ on a type $T$, with $T \models \phi$.

$$\phi \;:=\; \top \mid \bot \mid \phi \wedge \phi \mid \phi \vee \phi \mid$$
$$[\dagger a]\phi \mid \langle \dagger a \rangle \phi \mid \nu \mathbf{x}.\,\phi \mid \mathbf{x} \qquad \text{with } \dagger a := !a \mid ?a$$

## Example

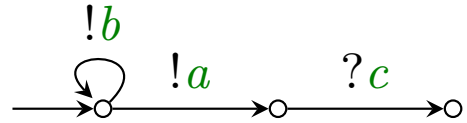
$\mathtt{rec}\,\mathbf{x}.(!a.\mathtt{end})\oplus(!b.\mathbf{x}) \;\models\; \nu \mathbf{x}.\,\langle !a \rangle \mathbf{x}$ ✗



# Modal $\mu$-calculus

A session type $T$ induces a **finite** LTS, e.g.,

$\mathtt{rec}\,\mathbf{x}.(!a.\,?c.\mathtt{end})\oplus(!b.\mathbf{x})$ gives

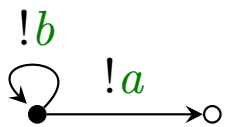



We can interpret a $\mu$-calculus formula $\phi$ on a type $T$, with $T \models \phi$.

$$\phi \;:=\; \top \mid \bot \mid \phi \wedge \phi \mid \phi \vee \phi \mid$$
$$[\dagger a]\phi \mid \langle \dagger a \rangle \phi \mid \nu \mathbf{x}.\,\phi \mid \mathbf{x} \qquad \text{with } \dagger a := !a \mid ?a$$

## Example

$\mathtt{rec}\,\mathbf{x}.(!a.\mathtt{end})\oplus(!b.\mathbf{x}) \;\models\; \nu \mathbf{x}.\,(\langle !a \rangle [\mathcal{A}]\bot \vee \langle !b \rangle \mathbf{x})$ ✓



# Characteristic formulae for subtyping session types

Given a (closed) session type $T$, $\boldsymbol{F}(T)$ is a $\mu$-calculus formula that characterises all the **supertypes** of $T$.

$$U \models \boldsymbol{F}(T) \iff T \leq U$$

$$\boldsymbol{F}(T) \stackrel{\text{def}}{=} \left\{ \right.$$

# Characteristic formulae for subtyping session types

Given a (closed) session type $T$, $\boldsymbol{F}(T)$ is a $\mu$-calculus formula that characterises all the **supertypes** of $T$.

$$U \models \boldsymbol{F}(\ !a \oplus !b\ ) \iff !a \oplus !b \leq U$$

$$\boldsymbol{F}(T) \stackrel{\text{def}}{=} \begin{cases} \bigwedge_{i \in I} \langle !a_i \rangle \boldsymbol{F}(T_i) & \text{if } T = \bigoplus_{i \in I} !a_i . T_i \end{cases}$$

# Characteristic formulae for subtyping session types

Given a (closed) session type $T$, $\boldsymbol{F}(T)$ is a $\mu$-calculus formula that characterises all the **supertypes** of $T$.

$$U \models \boldsymbol{F}(\ ?a \& ?b\ ) \iff ?a \& ?b \leq U$$

$$\boldsymbol{F}(T) \stackrel{\text{def}}{=} \begin{cases} \bigwedge_{i \in I} \langle !a_i \rangle \boldsymbol{F}(T_i) & \text{if } T = \bigoplus_{i \in I} !a_i . T_i \\ \bigwedge_{i \in I} [?a_i] \boldsymbol{F}(T_i) & \text{if } T = \&_{i \in I} ?a_i . T_i \end{cases}$$

# Characteristic formulae for subtyping session types

Given a (closed) session type $T$, $\boldsymbol{F}(T)$ is a $\mu$-calculus formula that characterises all the **supertypes** of $T$.

$$U \models \boldsymbol{F}(\ ?a \& ?b\ ) \iff ?a \& ?b \leq U$$

$$\boxed{?a \,\&\, ?b \not\leq \texttt{end}}$$

$$\boldsymbol{F}(T) \stackrel{\text{def}}{=} \begin{cases} \bigwedge_{i \in I} \langle !a_i \rangle \, \boldsymbol{F}(T_i) & \text{if } T = \bigoplus_{i \in I} !a_i . \, T_i \\ \bigwedge_{i \in I} [?a_i] \, \boldsymbol{F}(T_i) & \text{if } T = \&_{i \in I} ?a_i . \, T_i \\ \quad \wedge \bigvee_{i \in I} \langle ?a_i \rangle \top & \end{cases}$$

# Characteristic formulae for subtyping session types

Given a (closed) session type $T$, $\boldsymbol{F}(T)$ is a $\mu$-calculus formula that characterises all the **supertypes** of $T$.

$$U \models \boldsymbol{F}(\ ?a \& ?b\ ) \iff ?a \& ?b \leq U$$

$$\boxed{?a \,\&\, ?b \not\leq ?a \,\&\, ?c}$$

$$\boldsymbol{F}(T) \stackrel{\text{def}}{=} \begin{cases} \bigwedge_{i \in I} \langle !a_i \rangle \, \boldsymbol{F}(T_i) & \text{if } T = \bigoplus_{i \in I} !a_i . \, T_i \\ \bigwedge_{i \in I} [?a_i] \, \boldsymbol{F}(T_i) & \text{if } T = \&_{i \in I} ?a_i . \, T_i \\ \quad \wedge \bigvee_{i \in I} \langle ?a_i \rangle \top & \\ \quad \wedge [\neg \{ ?a_i \mid i \in I \}] \bot & \end{cases}$$

# Characteristic formulae for subtyping session types

Given a (closed) session type $T$, $\boldsymbol{F}(T)$ is a $\mu$-calculus formula that characterises all the **supertypes** of $T$.

$$U \models \boldsymbol{F}(\,\mathtt{end}\,) \iff \mathtt{end} \leq U$$

$$\boldsymbol{F}(T) \stackrel{\text{def}}{=} \begin{cases} \bigwedge_{i \in I} \langle !a_i \rangle \, \boldsymbol{F}(T_i) & \text{if } T = \bigoplus_{i \in I} !a_i . \, T_i \\ \bigwedge_{i \in I} [?a_i] \, \boldsymbol{F}(T_i) & \text{if } T = \mathbin{\&}_{i \in I} ?a_i . \, T_i \\ \quad \wedge \bigvee_{i \in I} \langle ?a_i \rangle \top & \\ \quad \wedge \, [\neg \{ ?a_i \mid i \in I \}] \bot & \\ [\mathcal{A}] \bot & \text{if } T = \mathtt{end} \end{cases}$$

# Characteristic formulae for subtyping session types

Given a (closed) session type $T$, $\boldsymbol{F}(T)$ is a $\mu$-calculus formula that characterises all the **supertypes** of $T$.

$$U \models \boldsymbol{F}(\,\mathtt{rec}\,\mathbf{x}.\,T'(\mathbf{x})\,) \iff \mathtt{rec}\,\mathbf{x}.\,T'(\mathbf{x}) \leq U$$

$$\boldsymbol{F}(T) \stackrel{\text{def}}{=} \begin{cases} \bigwedge_{i\in I} \langle !a_i \rangle\, \boldsymbol{F}(T_i) & \text{if } T = \bigoplus_{i\in I} !a_i.\, T_i \\ \bigwedge_{i\in I} [?a_i]\, \boldsymbol{F}(T_i) & \text{if } T = \&_{i\in I} ?a_i.\, T_i \\ \quad \wedge \bigvee_{i\in I} \langle ?a_i \rangle \top & \\ \quad \wedge\, [\neg\{?a_i \mid i \in I\}]\bot & \\ [\mathcal{A}]\bot & \text{if } T = \mathtt{end} \\ \nu\mathbf{x}.\ \boldsymbol{F}(T') & \text{if } T = \mathtt{rec}\,\mathbf{x}.T' \\ \mathbf{x} & \text{if } T = \mathbf{x} \end{cases}$$

# Characteristic formulae for subtyping session types

Given a (closed) session type $T$, $\boldsymbol{F}(T)$ is a $\mu$-calculus formula that characterises all the **supertypes** of $T$.

$$U \models \boldsymbol{F}(\,\mathtt{rec}\,\mathbf{x}.\,T'(\mathbf{x})\,) \iff \mathtt{rec}\,\mathbf{x}.\,T'(\mathbf{x}) \leq U$$

$$\boldsymbol{F}(T) \stackrel{\text{def}}{=} \begin{cases} \bigwedge_{i\in I} \langle !a_i \rangle\, \boldsymbol{F}(T_i) & \text{if } T = \bigoplus_{i\in I} !a_i.\, T_i \\ \bigwedge_{i\in I} [?a_i]\, \boldsymbol{F}(T_i) & \text{if } T = \&_{i\in I} ?a_i.\, T_i \\ \quad \wedge \bigvee_{i\in I} \langle ?a_i \rangle \top & \\ \quad \wedge\, [\neg\{?a_i \mid i \in I\}]\bot & \\ [\mathcal{A}]\bot & \text{if } T = \mathtt{end} \\ \nu\mathbf{x}.\ \boldsymbol{F}(T') & \text{if } T = \mathtt{rec}\,\mathbf{x}.T' \\ \mathbf{x} & \text{if } T = \mathbf{x} \end{cases}$$

## Theorem

*For all closed session types $T$ and $U$:* $\quad T \leq U \iff U \models \boldsymbol{F}(T)$.

# Parametric characteristic formula

In the paper, we give a parametric version of $\boldsymbol{F}(T, \maltese)$ (with $\maltese \in \{\oplus, \&\}$, such that, given a (closed) session type $T$:

- $\boldsymbol{F}(T, \oplus)$ is a $\mu$-calculus formula that characterises all the **supertypes** of $T$.
- $\boldsymbol{F}(T, \&)$ is a $\mu$-calculus formula that characterises all the **subtypes** of $T$.

## Theorem

1. $T \leq U \iff U \models \boldsymbol{F}(T, \oplus)$
2. $T \leq U \iff T \models \boldsymbol{F}(U, \&)$
3. $U \models \boldsymbol{F}(T, \oplus) \iff T \models \boldsymbol{F}(U, \&)$

# Safe system of session types

The subtyping $\leq$ is **sound** and **complete** wrt. safety for synchronous session types:

$$S \mid T \text{ is safe} \quad \Longleftrightarrow \quad S \leq \overline{T} \quad \Longleftrightarrow \quad T \leq \overline{S}$$

Thus we have:

**Theorem**

$$S \mid T \textit{ is safe} \quad \Longleftrightarrow \quad T \models \boldsymbol{F}(\overline{S}, \&) \quad \vee \quad \overline{T} \models \boldsymbol{F}(S, \oplus)$$

# Evaluation - Overview ($i$)

Four implementations of subtyping checker for session types:

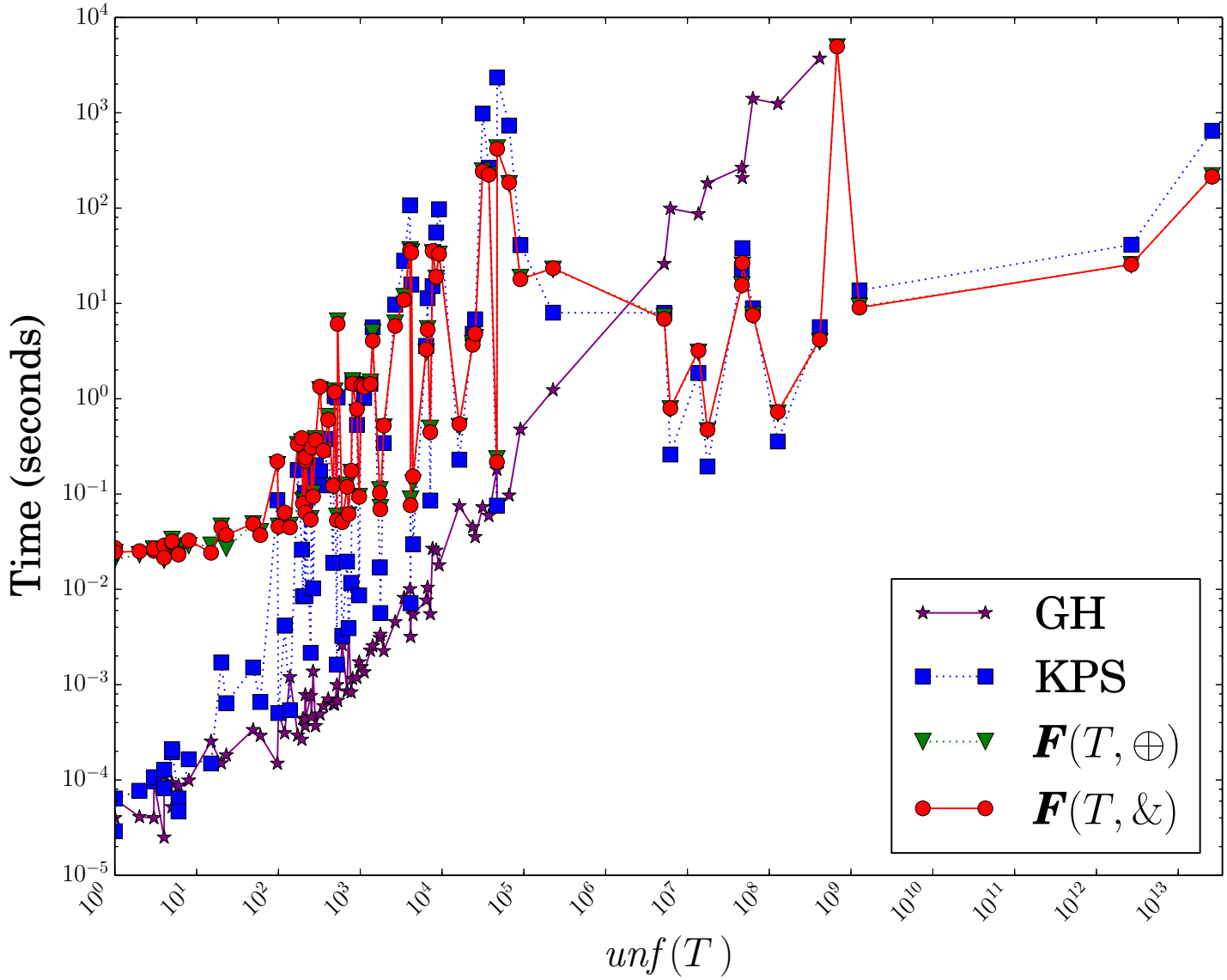
1. Original algorithm by S. Gay & M. Hole (**GH**)
   - Worst case **complexity: exponential**
2. Our adaptation of Kozen, Palsberg, & Schwartzbach ('95) (**KPS**)
   - *The algorithm reduces the problem of subtyping to checking the emptiness (of the language) of an automaton given by the product of two (session) types.*
   - Worst case **complexity: quadratic**
3. Model checking $T \models \boldsymbol{F}(U, \oplus)$
   - Worst case **complexity: quadratic**
4. Model checking $U \models \boldsymbol{F}(T, \&)$
   - Worst case **complexity: quadratic**

# Evaluation - Overview ($ii$)

- Implementation: **Haskell** and **mCRL2** model checker
- Data: randomly generated with Haskell's Quickcheck
- Benchmarks on pairs of (equivalent) session types
  - $T \leq T$
  - $T \leq unfold(T)$
- Measures:
  - $num(T)$ is the number of labels in $T$,
    e.g., $num(\mathtt{rec}\,\mathtt{x}.(!a.\,?c.\mathtt{end}) \oplus (!b.\mathtt{x})) = 3$.
  - $unf\,(T)$ returns the number of labels in the unfolding of $T$,
    e.g., $unf\,(\mathtt{rec}\,\mathtt{x}.(!a.\,?c.\mathtt{end}) \oplus 2(!b.\mathtt{x})) = 6$.

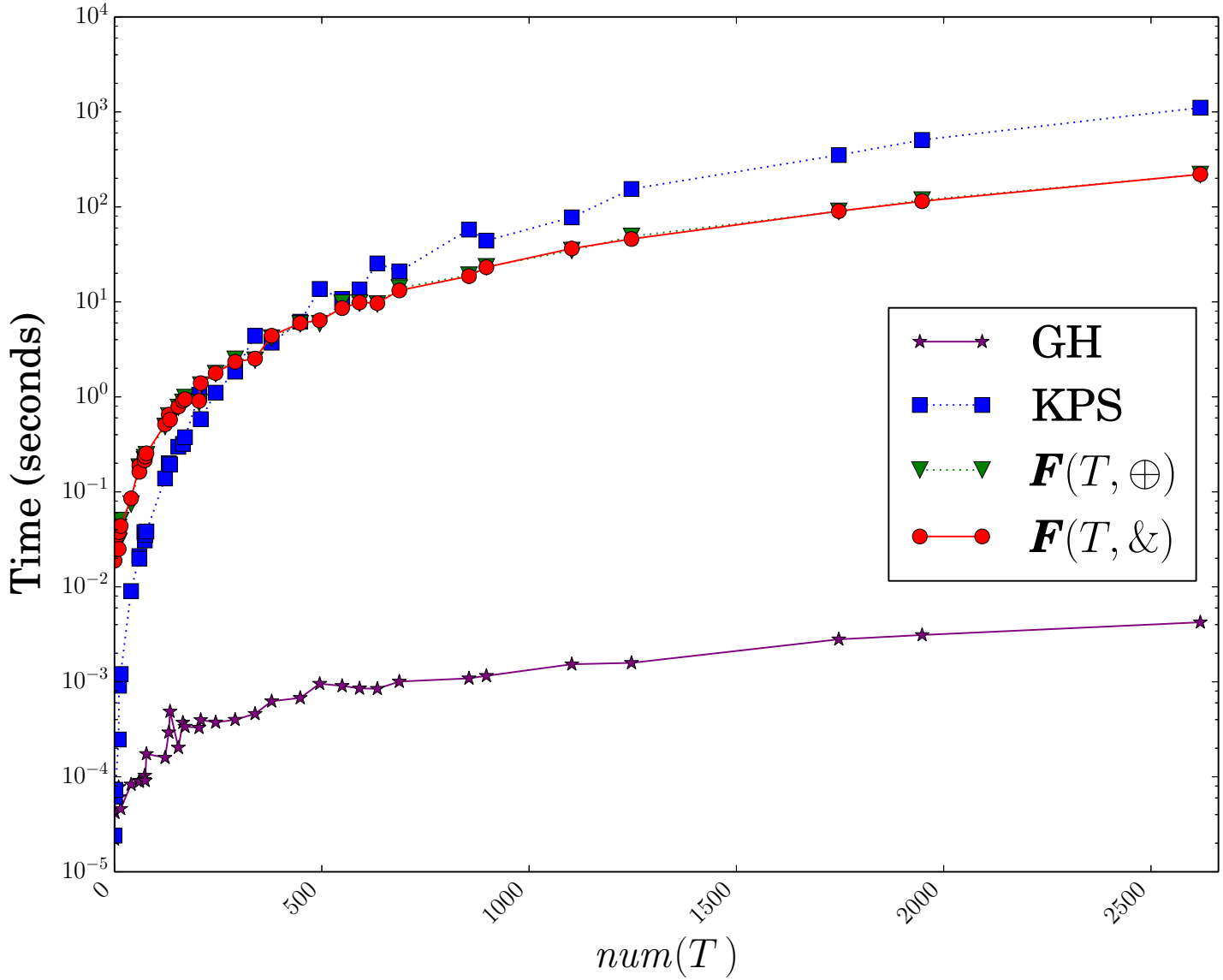
# Arbitrary session types (log. scale)

Results for $T \leq T$, with $T$ arbitrary.



# No recursive definition (lin. scale)

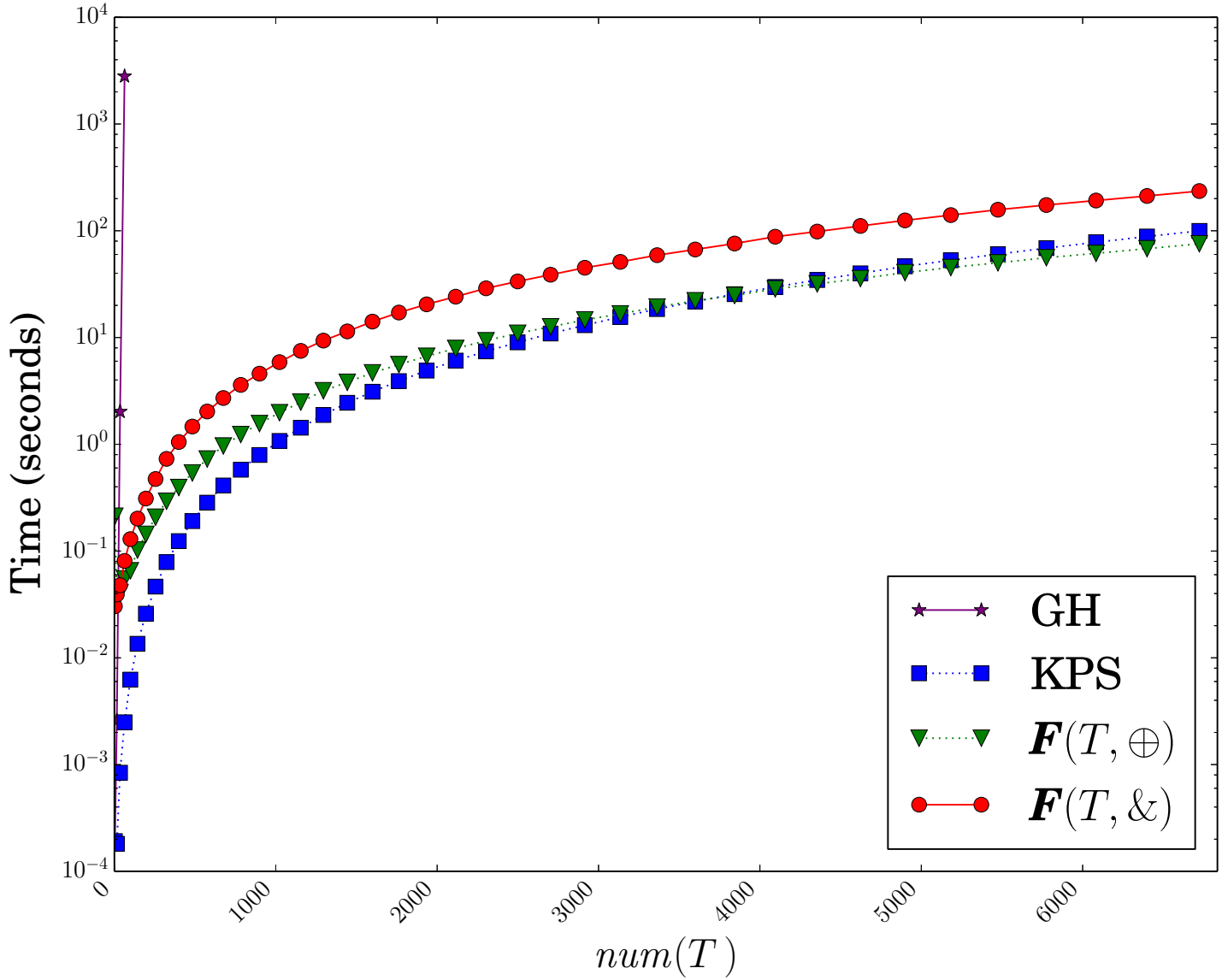
Results for $T \leq T$, with $T := \texttt{end} \mid \bigoplus_{i \in I} !a_i.\, T_i \mid \&_{i \in I} ?a_i.\, T_i$



# Super recursive ($i$) (lin. scale)

Results for $T \leq T$, with

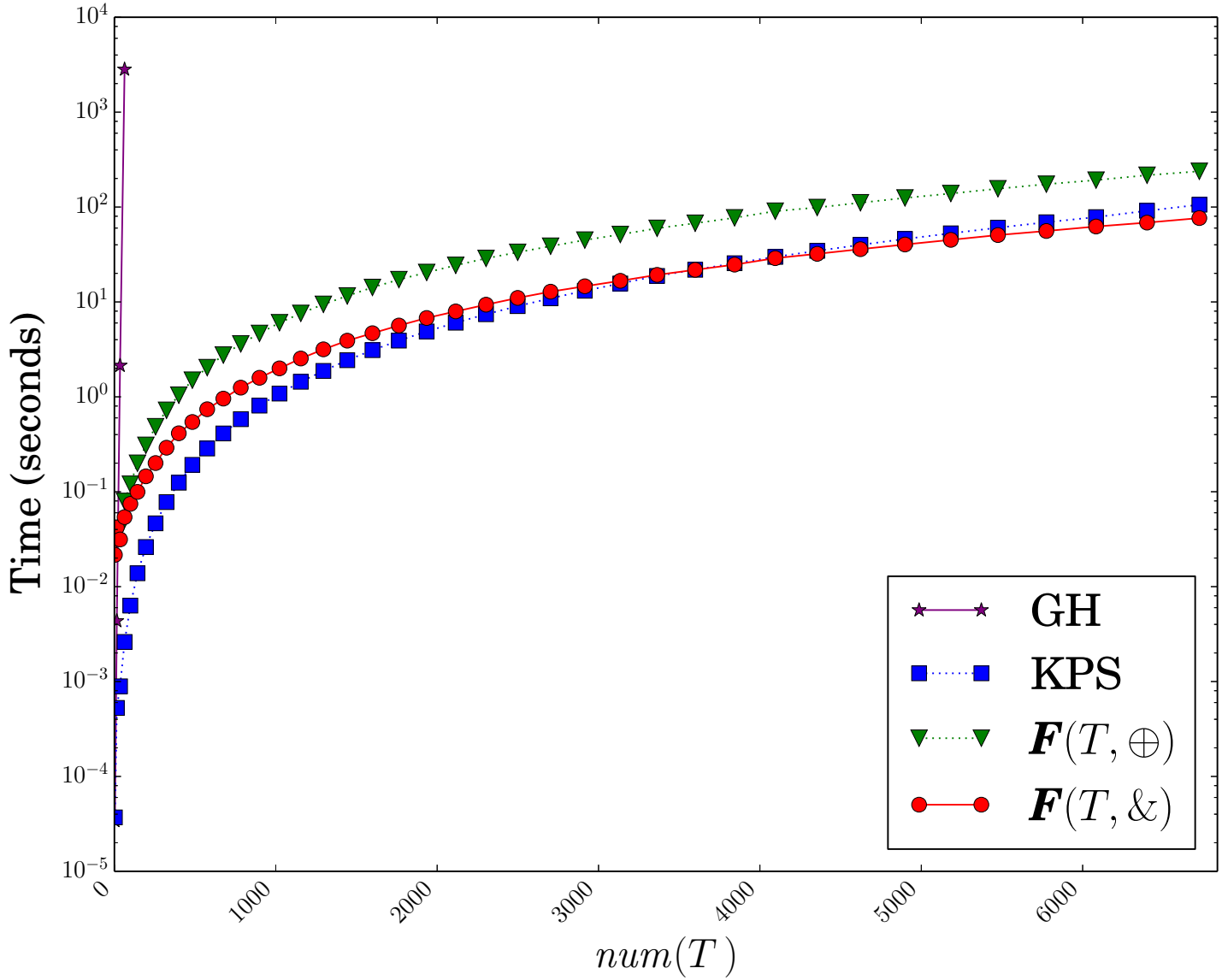
$T := \mathtt{rec}\,\mathbf{x}_1.!a_1 \ldots \mathtt{rec}\,\mathbf{x}_k.!a_k \{ \bigoplus_{1 \leq i \leq k} !a_i.\{\bigoplus_{1 \leq j \leq k} !a_j.\mathbf{x}_j\} \}$



# Super recursive ($ii$) (lin. scale)

Results for $T \leq T$, with

$T := \mathtt{rec}\,\mathbf{x}_1.?a_1 \ldots \mathtt{rec}\,\mathbf{x}_k.?a_k\,\{\,\&_{1\leq i\leq k}?a_i.\{\&_{1\leq j\leq k}?a_j.\mathbf{x}_j\}\,\}$



# Conclusions & Related Work

This work gives

- a first connection between model checking and session types,
- a reduction of subtyping to a model checking,
- a theoretical and experimental evaluation, and
- a similar construction for the recursive types for the $\lambda$-calculus,
- a tool: `http://bitbucket.org/julien-lange`

# Conclusions & Related Work

This work gives

- a first connection between model checking and session types,
- a reduction of subtyping to a model checking,
- a theoretical and experimental evaluation, and
- a similar construction for the recursive types for the $\lambda$-calculus,
- a tool: `http://bitbucket.org/julien-lange`

Some related work and references:

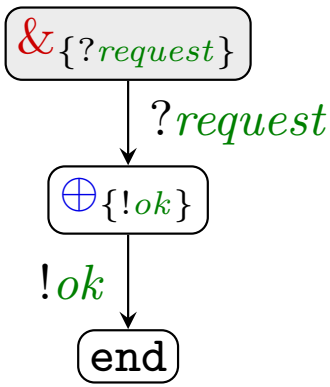
- B. Steffen. **Characteristic formulae** (1989)
- D. Kozen, J. Palsberg, & M. I. Schwartzbach. **Efficient recursive subtyping** (1995)
- S. J. Gay & M. Hole. **Types and subtypes for client-server interactions** (1999)
- R. Demangeon & K. Honda. **Full abstraction in a subtyped pi-calculus with linear types** (2011)
- T.-C. Chen, M. Dezani-Ciancaglini, & N. Yoshida. **On the preciseness of subtyping in session types** (2014)

# Thank you

Any questions?

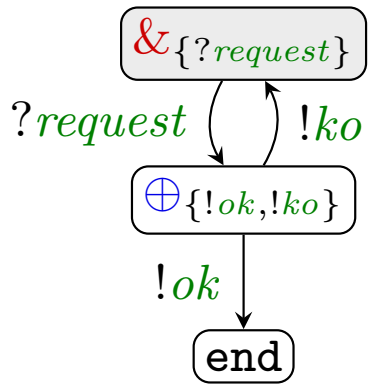
# Kozen, Palsberg, & Schwartzbach ('95) (**KPS**)

$$?request.!ok.\mathtt{end} \quad \leq \quad \mathtt{rec\,x}.?request.\{!ok.\mathtt{end} \oplus !ko.\mathtt{x}\}$$



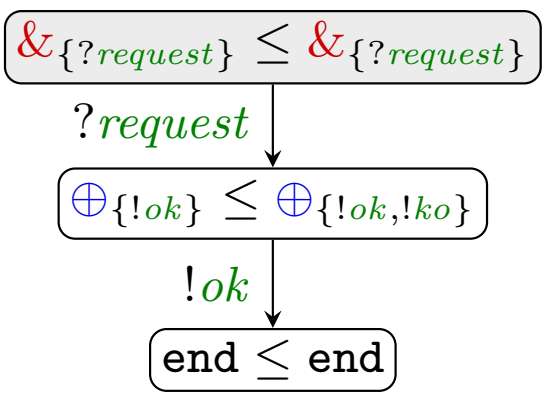

$\mathcal{M}(T_1)$

$\leq$



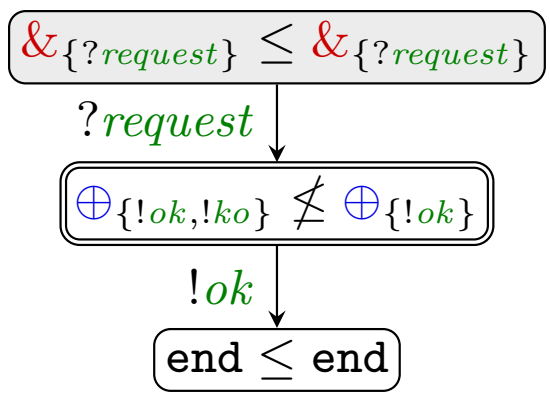

$\mathcal{M}(U_1)$

---




$\mathcal{M}(T_1) \blacktriangleleft \mathcal{M}(U_1)$

✓




$\mathcal{M}(U_1) \blacktriangleleft \mathcal{M}(T_1)$

✗