A symbolic labelled transition system for coinductive subtyping of $F_{\mu\leq}$ types

Alan Jeffrey
DePaul University

Extended Abstract

Abstract. *$F_{\leq}$ is a typed λ-calculus with subtyping and bounded polymorphism. Typechecking for $F_{\leq}$ is known to be undecidable, because the subtyping relation on types is undecidable. $F_{\mu\leq}$ is an extension of $F_{\leq}$ with recursive types. In this paper, we show how symbolic labelled transition system techniques from concurrency theory can be used to reason about subtyping for $F_{\mu\leq}$. We provide a symbolic labelled transition system for $F_{\mu\leq}$ types, together with an an appropriate notion of simulation, which coincides with the existing coinductive definition of subtyping. We then provide a 'simulation up to' technique for proving subtyping, for which there is a simple model checking algorithm. The algorithm is more powerful than the usual one for $F_{\leq}$, for example it terminates on Ghelli's canonical example of nontermination.*

1 Introduction

Symbolic labelled transition systems [11] have been used in concurrency theory to provide finite-state representations of infinite systems. They have been used to model-check systems with data dependencies, where the nïave state space exploration technique would produce an infinite state space, and so not terminate.

In this paper, we apply symbolic lts techniques to a new problem area: that of deciding subtyping for polymorphic λ-calculi.

Subtyping and polymorphism. Curien and Ghelli's [5] $F_{\leq}$ is a typed λ-calculus with bounded polymorphism and subtyping. It is based on Bruce and Longo's [2] development of Cardelli and Wegner's [3] *Fun* language.

The most interesting rule in $F_{\leq}$ is that for subtyping of polymorphic types:

$$\frac{\Gamma \vdash T_2 \leq T_1 \quad \Gamma, X \leq T_2 \vdash U_1 \leq U_2}{\Gamma \vdash (\forall X \leq T_1 \,.\, U_1) \leq (\forall X \leq T_2 \,.\, U_2)} \text{ (Full } F_{\leq}\text{)}$$

This is a stronger rule than the rule used in *Fun*, which is:

$$\frac{\Gamma, X \leq T \vdash U_1 \leq U_2}{\Gamma \vdash (\forall X \leq T \,.\, U_1) \leq (\forall X \leq T \,.\, U_2)} \text{ (Kernel } F_{\leq}\text{)}$$

It is routine to develop an algorithm to check the subtyping property of Kernel $F_{\leq}$, but subtyping for Full $F_{\leq}$ has turned out to be surprisingly complex. Curien and Ghelli [5] gave an algorithm for checking subtyping, with a correctness proof provided by Ghelli [7]. Later, Ghelli [9] showed that this algorithm is not guaranteed to terminate. Pierce [14] showed that Ghelli's example of nontermination can be generalized to code a Turing machine, and so subtyping (and hence typechecking) for $F_{\leq}$ is undecidable.

Subtyping and recursive types. Recursive types are a common programming language feature, typified by ML's `datatype` construct. Amadio and Cardelli [17] investigated the relationship between subtyping and recursive types. Brand and Henglein [1] reformulated subtyping in terms of coinductive relations on types, which we will use here. The coinductive presentation of type systems for subtyping in the presence of recursive types has been used by Pierce and Sangiorgi [16] for the π-calculus, Turner [20] for Pict and Sewell [19] for a distributed π-calculus. A good introduction is by Gapeyev, Levin and Pierce [6].

Ghelli [8] has investigated the relationship between subtyping, recursive types and polymorphic types, in the recursive extension to $F_{\leq}$, called $F_{\mu\leq}$. He has shown a number of surprising results: adding recursion to $F_{\leq}$ is not conservative, and $F_{\mu\leq}$ does not satisfy the transitivity elimination property. These results are for the inductive definition of subtyping, however, where here we look at the coinductive definition, which is much better behaved. Colazzo and Ghelli have provided an algorithm for deciding subtyping of Kernel $F_{\mu\leq}$ [4]: much of this paper is based on that algorithm.

Symbolic labelled transition systems. Labelled transition systems are a form of nondeterministic automaton, where all states are considered to be accepting states. They were proposed by Milner [12, 13] as an appropriate model for concurrent systems. They have since been used to model higher-order computation, for example Gordon's [10] lts model of the simply-typed λ-calculus.

One problem with lts models is that they can produce infinite models of systems which should be finite-state. For example, the process defined:

$$P = \mathsf{in}\ (x : \mathsf{int}); \mathsf{out}\ (x+1); P$$

has transitions:

$$(P) \xrightarrow{\mathsf{in}\ (n)} (\mathsf{out}\ (n+1); P) \xrightarrow{\mathsf{out}\ (n+1)} (P)$$

for every integer n and so is infinite-state. Hennessy and Lin [11] proposed using *symbolic* labelled transition systems as an appropriate finitary representation. A symbolic lts includes free variables, so rather than having nodes being closed processes, and edges labelled with closed expressions, the nodes are processes together with their free variables, and the edges are labelled with open expressions. For example:

$$(\vdash P) \xrightarrow{\mathsf{in}\ (x:\mathsf{int})} (x : \mathsf{int} \vdash \mathsf{out}\ (x+1); P) \xrightarrow{\mathsf{out}\ (x+1)} (x : \mathsf{int} \vdash P)$$

Unfortunately, this system is still infinite-state, since the context can grow unboundedly:

$$\begin{tikzcd}
(\vdash P) \arrow[r, "\mathsf{in}\ (x:\mathsf{int})"] & (x : \mathsf{int} \vdash \mathsf{out}\ (x+1); P) \arrow[dl, "\mathsf{out}\ (x+1)"'] \\
(x : \mathsf{int} \vdash P) \arrow[r, "\mathsf{in}\ (x':\mathsf{int})"] & (x : \mathsf{int}, x' : \mathsf{int} \vdash \mathsf{out}\ (x'+1); P) \arrow[dl, "\mathsf{out}\ (x'+1)"'] \\
(x : \mathsf{int}, x' : \mathsf{int} \vdash P) & \cdots
\end{tikzcd}$$

For this reason, symbolic techniques often work 'up to garbage collection' where unneeded free variables can be removed from the context. For example, the above process can be given a finite symbolic representation as:

$$\begin{tikzcd}
(\vdash P) \arrow[rr, "\mathsf{in}\ (x:\mathsf{int})"] & & (x : \mathsf{int} \vdash \mathsf{out}\ (x+1); P) \arrow[dl, "\mathsf{out}\ (x+1)"] \\
& (x : \mathsf{int} \vdash P) \arrow[ul, "\mathsf{gc}\ (x:\mathsf{int})"] &
\end{tikzcd}$$

Symbolic lts's have been used to provide finite-state representations of systems that would otherwise be infinite-state.

Contributions of this paper. In this paper, we apply the techniques of symbolic labelled transition systems to the problem of subtyping $F_{\mu\leq}$. In particular, we:

- Give an alternative characterization of subtyping for $F_{\mu\leq}$, as *polar simulation* for an appropriate symbolic lts.

- Use a variant of Milner and Sangiorgi's [18] *bisimulation up to* method to give a sound proof technique for subtyping.

- Provide an algorithm for finding an appropriate polar simulation, if one exists.

- Show that the algorithm is partially correct: if it terminates, it does so with the right answer.

- Show that the algorithm is strictly more powerful than the standard algorithm for $F_{\leq}$, and at least as powerful as Colazzo and Ghelli's algorithm for Kernel $F_{\mu\leq}$.

Acknowledgements. I would like to thank Benjamin Pierce, James Riely and Peter Sewell for useful discussions about this material. Donald Knuth's TEX typesetting system, Leslie Lamport *et al.*'s LATEX document markup language, and Paul Taylor's diagrams package were used in the preparation of this paper.

2 The type system of $F_{\mu\leq}$

In this section, we review the types system used in Ghelli's [8] $F_{\mu\leq}$. There are some minor syntactic differences between the types presented here and Ghelli's, but they are equally expressive. We have added type constants such as int and real to the language, to make examples clearer, they are not required for any of the technical development.

Let K range over a finite collection of type constants, such as int and real. The syntax of types is given:

$$T, U, V \quad ::= \quad T \to U \mid \mathsf{Top} \mid K \mid \forall X \leq T \,.\, U \mid \mu^+ X \,.\, T \mid X$$

Define the *free variables* of a type as:

$$\mathsf{fv}\,(T) \quad = \quad \mathsf{fv}^+\,(T) \cup \mathsf{fv}^-\,(T)$$

where the *polarized free variables* are:

$$\begin{array}{rcl}
\mathsf{fv}^\pm\,(T \to U) & = & \mathsf{fv}^\mp\,(T) \cup \mathsf{fv}^\pm\,(U) \\
\mathsf{fv}^\pm\,(\mathsf{Top}) & = & \emptyset \\
\mathsf{fv}^\pm\,(K) & = & \emptyset \\
\mathsf{fv}^\pm\,(\forall X \leq T \,.\, U) & = & \mathsf{fv}^\mp\,(T) \cup (\mathsf{fv}^\pm\,(U) \setminus \{X\}) \\
\mathsf{fv}^\pm\,(\mu^+ X \,.\, T) & = & \mathsf{fv}^\pm\,(T) \setminus \{X\} \\
\mathsf{fv}^+\,(X) & = & \{X\} \\
\mathsf{fv}^-\,(X) & = & \emptyset
\end{array}$$

A *type context* is a sequence of variables with type bounds:

$$\Gamma, \Delta \quad ::= \quad X_1 \leq T_1, \ldots, X_n \leq T_n$$

where we ignore the order of bindings. The *domain* of a context $\mathsf{dom}\,(\Gamma)$ is defined:

$$\mathsf{dom}\,(X_1 \leq T_1, \ldots, X_n \leq T_n) \quad = \quad \{X_1, \ldots, X_n\}$$

When $X \in \mathsf{dom}(\Gamma)$ we define $\Gamma(X)$ as:

$$(\Gamma, X \leq T)(X) \quad = \quad T$$

The *well-formed context* judgment $\Gamma \vdash \diamond$ is defined:

$$\frac{}{\emptyset \vdash \diamond} \qquad \frac{\Gamma \vdash T}{\Gamma, X \leq T \vdash \diamond}[X \notin \mathsf{dom}(\Gamma)]$$

where the *well-formed type* judgment $\Gamma \vdash T$ is defined:

$$\frac{\Gamma \vdash T \quad \Gamma \vdash U}{\Gamma \vdash T \to U} \qquad \frac{\Gamma \vdash \diamond}{\Gamma \vdash \mathsf{Top}} \qquad \frac{\Gamma \vdash \diamond}{\Gamma \vdash K} \qquad \frac{\Gamma, X \leq T \vdash U}{\Gamma \vdash \forall X \leq T \,.\, U}$$

$$\frac{\Gamma, X \leq T \vdash \diamond}{\Gamma, X \leq T \vdash X} \qquad \frac{\Gamma, X \leq \mathsf{Top} \vdash T}{\Gamma \vdash \mu^+ X \,.\, T}[X \notin \mathsf{fv}^-(T), T \neq Y]$$

Note that we have required X to occur positively in T in any recursive type $\mu^+ X \,.\, T$, and that we cannot form recursive types of the form $\mu^+ X \,.\, Y$. These restrictions do not limit the expressive power of the type system, since for any $T(X)$ we can find $T'(X, X')$ such that:

$$T(X) = T'(X, X)$$
$$X \notin \mathsf{fv}^-(T'(X, X')) \quad X' \notin \mathsf{fv}^+(T'(X, X'))$$

then we can define:

$$\mu X \,.\, T(X) \quad = \quad \mu^+ X_1 \,.\, T'(X_1, \mu^+ X_2 \,.\, T'(X_2, X_1))$$

and we can give a greatest fixed point semantics for $\mu X \,.\, T$ as:

$$\mu X \,.\, Y \quad = \quad \begin{cases} \mathsf{Top} & \text{if } X = Y \\ Y & \text{otherwise} \end{cases}$$

We define α-equivalence on well-formed types as (when $Y \notin \mathsf{dom}(\Gamma)$):

$$(\Gamma, X \leq U \vdash T) \overset{Y/X}{=\!=} (\Gamma[Y/X], Y \leq U \vdash T[Y/X])$$

We assume an ordering $K_1 \leq K_2$ on type constants, for example $\mathsf{int} \leq \mathsf{real}$. This is extended to an *inductive subtyping* judgment $\Gamma \vdash T_1 \leq T_2$ defined:

$$\frac{}{\Gamma \vdash T \leq T} \qquad \frac{\Gamma \vdash T_2 \leq T_1 \quad \Gamma \vdash U_1 \leq U_2}{\Gamma \vdash (T_1 \to U_1) \leq (T_2 \to U_2)}$$

$$\frac{}{\Gamma \vdash T \leq \mathsf{Top}} \qquad \frac{K_1 \leq K_2}{\Gamma \vdash K_1 \leq K_2}$$

$$\frac{\Gamma \vdash T_2 \leq T_1 \quad \Gamma, X \leq T_2 \vdash U_1 \leq U_2}{\Gamma \vdash (\forall X \leq T_1 \,.\, U_1) \leq (\forall X \leq T_2 \,.\, U_2)} \qquad \frac{\Gamma \vdash \Gamma(X) \leq T}{\Gamma \vdash X \leq T}$$

$$\frac{\Gamma \vdash T_1[(\mu^+ X \,.\, T_1)/X] \leq T_2}{\Gamma \vdash (\mu^+ X \,.\, T_1) \leq T_2} \qquad \frac{\Gamma \vdash T_1 \leq T_2[(\mu^+ X \,.\, T_2)/X]}{\Gamma \vdash T_1 \leq (\mu^+ X \,.\, T_2)}$$

A *well-formed relation on types* $\mathcal{R}$ is a relation $\mathcal{R}$ on well-formed types $\Gamma \vdash T$ such that if $(\Gamma_1 \vdash T_1) \,\mathcal{R}\, (\Gamma_2 \vdash T_2)$ then $\Gamma_1 = \Gamma_2$. We shall often write $\Gamma \vDash T_1 \,\mathcal{R}\, T_2$ when $(\Gamma \vdash T_1) \,\mathcal{R}\, (\Gamma \vdash T_2)$. For example, the inductive subtyping relation $\leq$ gives a well-formed relation on types:

$$\Gamma \vDash T \leq U \quad \text{iff} \quad \Gamma \vdash T \leq U$$

We regard well-formed relations on types up to α-equivalence, so we can complete the diagram:

$$\begin{array}{ccc} (\Gamma \vdash T) \overset{\mathcal{R}}{\longleftrightarrow} (\Gamma \vdash U) & & (\Gamma \vdash T) \overset{\mathcal{R}}{\longleftrightarrow} (\Gamma \vdash U) \\ {\scriptstyle Y/X}\Big\| \qquad\qquad {\scriptstyle Y/X}\Big\| & \text{as} & {\scriptstyle Y/X}\Big\| \qquad\qquad {\scriptstyle Y/X}\Big\| \\ (\Gamma' \vdash T') \qquad (\Gamma' \vdash U') & & (\Gamma' \vdash T') \overset{\mathcal{R}}{\longleftrightarrow} (\Gamma' \vdash U') \end{array}$$

A well-formed relation on types $\mathcal{R}$ is *sound for subtyping* if, for every instantiated subtyping rule:

$$\frac{\Gamma_1 \vdash T_1 \leq U_1 \quad \cdots \quad \Gamma_n \vdash T_n \leq U_n}{\Gamma \vdash T \leq U}$$

we have:

if $\Gamma_1 \vDash T_1 \,\mathcal{R}\, U_1$ and $\ldots$ and $\Gamma_n \vDash T_n \,\mathcal{R}\, U_n$ then $\Gamma \vDash T \,\mathcal{R}\, U$

A well-formed relation on types $\mathcal{R}$ is *consistent with subtyping* if it is sound for subtyping, and whenever $\Gamma \vDash T \,\mathcal{R}\, U$ we can find an instantiated subtyping rule:

$$\frac{\Gamma_1 \vdash T_1 \leq U_1 \quad \cdots \quad \Gamma_n \vdash T_n \leq U_n}{\Gamma \vdash T \leq U}$$

such that:

$$\Gamma_1 \vDash T_1 \,\mathcal{R}\, U_1 \text{ and } \ldots \text{ and } \Gamma_n \vDash T_n \,\mathcal{R}\, U_n$$

Let the *coinductive subtyping* relation $\sqsubseteq$ be the largest relation consistent with subtyping.

Proposition 1 *$\leq$ is the smallest relation consistent with subtyping, and so $\leq \subseteq \sqsubseteq$.*

3 Motivation for the symbolic lts semantics for $F_{\mu\leq}$

This paper provides an alternative characterization of subtyping for $F_{\mu\leq}$, using a symbolic labelled transition system. By recasting coinductive subtyping as an lts, it is possible to use existing tools from concurrency theory, notably Milner and Sangiorgi's *bisimulation up to* technique.

The lts has well-formed types as nodes, and edges which reflect the structure of the type. For example, the Top type has no transitions:

$$(\Gamma \vdash \mathsf{Top}) \overset{\alpha}{\not\longrightarrow} (\Gamma' \vdash T')$$

and the type constants have transitions with their name:

$$(\Gamma \vdash \mathsf{int}) \xrightarrow{\mathsf{int}} (\Gamma \vdash \mathsf{Top}) \quad (\Gamma \vdash \mathsf{real}) \xrightarrow{\mathsf{real}} (\Gamma \vdash \mathsf{Top})$$

We can think of the subtyping relation as a *simulation* [13] relation: if T is a supertype of U then any transition of T must have a matching transition from U. For example we can complete the following diagram:

$$\begin{tikzcd}
(\vdash \mathsf{real}) \arrow[r, leftrightarrow, "\gtrsim"] \arrow[d, "\mathsf{real}"'] & (\vdash \mathsf{int}) \\
(\vdash \mathsf{Top}) &
\end{tikzcd}
\quad \text{as} \quad
\begin{tikzcd}
(\vdash \mathsf{real}) \arrow[r, leftrightarrow, "\gtrsim"] \arrow[d, "\mathsf{real}"'] & (\vdash \mathsf{int}) \arrow[d, Rightarrow, "\widehat{\mathsf{real}}"] \\
(\vdash \mathsf{Top}) \arrow[r, leftrightarrow, "\gtrsim"] & (\vdash \mathsf{Top})
\end{tikzcd}$$

We define the 'matching transition relation' $\overset{\widehat{\alpha}}{\Longrightarrow}$ formally in Section 4, for the moment we will just say that it includes $\xrightarrow{\alpha}$, but also includes:

$$(\Gamma \vdash \mathsf{int}) \overset{\widehat{\mathsf{real}}}{\Longrightarrow} (\Gamma \vdash \mathsf{Top})$$

This notion of a 'matching transition relation' is standard in process calculi, where it is used to define weak bisimulation [13]. In general, a *simulation* $\gtrsim$ is a well-formed relation on types where we can complete the diagram:

$$\begin{tikzcd}
(\Gamma \vdash T_1) \arrow[r, leftrightarrow, "\gtrsim"] \arrow[d, "\alpha"'] & (\Gamma \vdash T_2) \\
(\Gamma' \vdash T_1') &
\end{tikzcd}
\quad \text{as} \quad
\begin{tikzcd}
(\Gamma \vdash T_1) \arrow[r, leftrightarrow, "\gtrsim"] \arrow[d, "\alpha"'] & (\Gamma \vdash T_2) \arrow[d, Rightarrow, "\widehat{\alpha}"] \\
(\Gamma' \vdash T_1') \arrow[r, leftrightarrow, "\gtrsim"] & (\Gamma' \vdash T_2')
\end{tikzcd}$$

Function types have domain and codomain transitions:

$$\begin{tikzcd}
& (\Gamma \vdash T \to U) \arrow[dl, "\mathsf{dom}"'] \arrow[dr, "\mathsf{cod}"] & \\
(\Gamma \vdash T) & & (\Gamma \vdash U)
\end{tikzcd}$$

Since function types are contravariant in their first argument and covariant in their second argument, we introduce *polarity* to labels: dom is negative polarity, and cod is positive polarity. This is important when we consider the subtyping relation, for example:

$$\begin{tikzcd}
(\vdash \mathsf{int} \to \mathsf{real}) \arrow[rrr, leftrightarrow, "\gtrsim"] \arrow[dd, "\mathsf{cod}"'] \arrow[dr, "\mathsf{dom}"] & & & (\vdash \mathsf{real} \to \mathsf{int}) \arrow[dl, Rightarrow, "\widehat{\mathsf{dom}}"'] \arrow[dd, Rightarrow, "\widehat{\mathsf{cod}}"] \\
& (\vdash \mathsf{int}) \arrow[r, leftrightarrow, "\lesssim"] & (\vdash \mathsf{real}) & \\
(\vdash \mathsf{real}) \arrow[rrr, leftrightarrow, "\gtrsim"] & & & (\vdash \mathsf{int})
\end{tikzcd}$$

Note that after a dom transition, the subtyping relation is inverted, but after a cod transition, it is not. A well-formed relation $\gtrsim$ is a *polar simulation* if it acts as a simulation on positive labels, and on negative labels we can complete the diagram:

$$\begin{tikzcd}
(\Gamma \vdash T_1) \arrow[r, leftrightarrow, "\gtrsim"] \arrow[d, "\alpha^-"'] & (\Gamma \vdash T_2) \\
(\Gamma' \vdash T_1') &
\end{tikzcd}
\quad \text{as} \quad
\begin{tikzcd}
(\Gamma \vdash T_1) \arrow[r, leftrightarrow, "\gtrsim"] \arrow[d, "\alpha^-"'] & (\Gamma \vdash T_2) \arrow[d, Rightarrow, "\widehat{\alpha^-}"] \\
(\Gamma' \vdash T_1') \arrow[r, leftrightarrow, "\lesssim"] & (\Gamma' \vdash T_2')
\end{tikzcd}$$

To cope with recursive types, we allow *silent actions* τ, where recursive types can silently unwind:

$$(\Gamma \vdash \mu^+ X\,.\,T) \xrightarrow{\tau} (\Gamma \vdash T[\mu^+ X\,.\,T/X])$$

For example, if we define:

$$T = \mu^+ X\,.\,\mathsf{int} \to X \qquad U = \mu^+ Y\,.\,\mathsf{int} \to \mathsf{real} \to Y$$

then we have a polar simulation for $T \gtrsim U$, since we define the matching transition relation to ignore τ actions:

$$\begin{tikzcd}
& (\vdash \mathsf{int}) \arrow[r, leftrightarrow, "\lesssim"] & (\vdash \mathsf{int}) & \\
(\vdash \mathsf{int} \to T) \arrow[ur, "\mathsf{dom}"] \arrow[dr, "\mathsf{cod}"'] \arrow[rrr, leftrightarrow, "\gtrsim"] & & & (\vdash \mathsf{int} \to \mathsf{real} \to U) \arrow[ul, Rightarrow, "\widehat{\mathsf{dom}}"'] \arrow[dl, Rightarrow, "\widehat{\mathsf{cod}}"] \\
& (\vdash T) \arrow[r, leftrightarrow, "\gtrsim"] \arrow[d, "\tau"'] & (\vdash \mathsf{real} \to U) \arrow[d, Rightarrow, "\widehat{\tau}"] & \\
& (\vdash \mathsf{int} \to T) \arrow[r, leftrightarrow, "\gtrsim"] \arrow[d, "\mathsf{dom}"] \arrow[ddl, "\mathsf{cod}"'] & (\vdash \mathsf{real} \to U) \arrow[d, Rightarrow, "\widehat{\mathsf{dom}}"'] \arrow[ddr, Rightarrow, "\widehat{\mathsf{cod}}"] & \\
& (\vdash \mathsf{int}) \arrow[r, leftrightarrow, "\lesssim"] & (\vdash \mathsf{real}) & \\
(\vdash T) \arrow[uuuu, "\tau"] \arrow[rrr, leftrightarrow, "\gtrsim"] & & & (\vdash U) \arrow[uuuu, Rightarrow, "\widehat{\tau}"']
\end{tikzcd}$$

Since we are giving a semantics for types with free variables, we need to give variables transitions: they can either announce themselves, or behave like their bound:

$$\begin{tikzcd}
& (\Gamma \vdash X) \arrow[dl, "X"'] \arrow[dr, "\tau"] & \\
(\Gamma \vdash \mathsf{Top}) & & (\Gamma \vdash \Gamma(X))
\end{tikzcd}$$

For example, $X \leq \mathsf{int} \vDash \mathsf{int} \gtrsim X$ since:

$$\begin{tikzcd}
(X \leq \mathsf{int} \vdash \mathsf{int}) \arrow[r, leftrightarrow, "\gtrsim"] \arrow[d, "\mathsf{int}"'] & (X \leq \mathsf{int} \vdash X) \arrow[d, Rightarrow, "\widehat{\mathsf{int}}"] \\
(X \leq \mathsf{int} \vdash \mathsf{Top}) \arrow[r, leftrightarrow, "\gtrsim"] & (X \leq \mathsf{int} \vdash \mathsf{Top})
\end{tikzcd}$$

Finally, we are left with the meat of the problem: modelling bounded polymorphism. Modelling Kernel $F_{\mu\le}$ is not too difficult, we just add transitions which reveal the structure of a polymorphic type:

$$
\begin{tikzcd}
& (\Gamma \vdash \forall X \le T \,.\, U) \arrow[dl, "\mathsf{bound}"'] \arrow[dr, "\forall X \le T"] & \\
(\Gamma \vdash T) & & (\Gamma, X \le T \vdash U)
\end{tikzcd}
$$

For example, $\vDash (\forall X \le \mathsf{int} \,.\, \mathsf{int}) \gtrsim (\forall X \le \mathsf{int} \,.\, X)$ since:

$$
\begin{tikzcd}
(\vdash \forall X \le \mathsf{int} \,.\, \mathsf{int}) \arrow[rrr, leftrightarrow, "\gtrsim"] \arrow[dd, "\forall X \le \mathsf{int}"'] \arrow[dr, "\mathsf{bound}"] & & & (\vdash \forall X \le \mathsf{int} \,.\, X) \arrow[dl, Rightarrow, "\widehat{\mathsf{bound}}"'] \arrow[dd, Rightarrow, "\widehat{\forall X \le \mathsf{int}}"] \\
& (\vdash \mathsf{int}) \arrow[r, leftrightarrow, "\lesssim"] & (\vdash \mathsf{int}) & \\
(X \le \mathsf{int} \vdash \mathsf{int}) \arrow[rrr, leftrightarrow, "\gtrsim"] & & & (X \le \mathsf{int} \vdash X)
\end{tikzcd}
$$

In order to model Full $F_{\mu\le}$, however, we have to allow the bound of a polymorphic type to vary. We do this by adding an additional transition to the matching transition relation:

$$
(\Gamma \vdash \forall X \le T \,.\, U) \overset{\widehat{\forall X \le V}}{\Longrightarrow} (\Gamma, X \le V \vdash U)
$$

For example, $\vDash (\forall X \le \mathsf{int} \,.\, \mathsf{int}) \gtrsim (\forall X \le \mathsf{real} \,.\, X)$ since:

$$
\begin{tikzcd}
(\vdash \forall X \le \mathsf{int} \,.\, \mathsf{int}) \arrow[rrr, leftrightarrow, "\gtrsim"] \arrow[dd, "\forall X \le \mathsf{int}"'] \arrow[dr, "\mathsf{bound}"] & & & (\vdash \forall X \le \mathsf{real} \,.\, X) \arrow[dl, Rightarrow, "\widehat{\mathsf{bound}}"'] \arrow[dd, Rightarrow, "\widehat{\forall X \le \mathsf{int}}"] \\
& (\vdash \mathsf{int}) \arrow[r, leftrightarrow, "\lesssim"] & (\vdash \mathsf{real}) & \\
(X \le \mathsf{int} \vdash \mathsf{int}) \arrow[rrr, leftrightarrow, "\gtrsim"] & & & (X \le \mathsf{int} \vdash X)
\end{tikzcd}
$$

In general, since bound is a negative label, it is easy to see that the following diagram models the Full $F_{\mu\le}$ rule for subtyping bounded polymorphism:

$$
\begin{tikzcd}
(\Gamma \vdash \forall X \le T_2 \,.\, U_2) \arrow[rrr, leftrightarrow, "\gtrsim"] \arrow[dd, "\forall X \le T_2"'] \arrow[dr, "\mathsf{bound}"] & & & (\Gamma \vdash \forall X \le T_1 \,.\, U_1) \arrow[dl, Rightarrow, "\widehat{\mathsf{bound}}"'] \arrow[dd, Rightarrow, "\widehat{\forall X \le T_2}"] \\
& (\Gamma \vdash T_2) \arrow[r, leftrightarrow, "\lesssim"] & (\Gamma \vdash T_1) & \\
(\Gamma, X \le T_2 \vdash U_2) \arrow[rrr, leftrightarrow, "\gtrsim"] & & & (\Gamma, X \le T_2 \vdash U_1)
\end{tikzcd}
$$

As a final example, we consider Ghelli's [9] example of non-termination of the standard algorithm for $F_\le$ subtyping:

$$
G \quad = \quad \forall X \,.\, \neg(\forall Y \le X \,.\, \neg Y)
$$

where we write $\neg T$ as shorthand for $T \to \mathsf{Top}$, and $\forall X \,.\, T$ as shorthand for $\forall X \le \mathsf{Top} \,.\, T$. Ghelli's example is to verify:

$$
X_0 \le G \vDash (\forall X_1 \le X_0 \,.\, \neg X_1) \gtrsim X_0
$$

If we define:

$$
\begin{aligned}
\Gamma_n &= X_0 \le G, X_1 \le X_0, \ldots, X_n \le X_{n-1} \\
G_n &= \forall X_{n+1} \le X_n \,.\, \neg X_{n+1}
\end{aligned}
$$

then $\Gamma_n \vDash G_n \gtrsim X_n$ for every n since:

$$
\begin{tikzcd}
{} \arrow[d, Rightarrow] & & \vdots & & {} \arrow[d] \\
(\Gamma_n \vdash G_n) \arrow[rrrr, leftrightarrow, "\gtrsim"] \arrow[dd, "\forall X_{n+1} \le X_n"'] \arrow[dr, "\mathsf{bound}"] & & & & (\Gamma_n \vdash X_n) \arrow[dl, Rightarrow, "\widehat{\mathsf{bound}}"'] \arrow[dd, Rightarrow, "\widehat{\forall X_{n+1} \le X_n}"] \\
& (\Gamma \vdash X_n) \arrow[rr, leftrightarrow, "\lesssim"] & & (\Gamma \vdash \mathsf{Top}) & \\
(\Gamma_{n+1} \vdash \neg X_{n+1}) \arrow[rrrr, leftrightarrow, "\gtrsim"] \arrow[d, "\mathsf{dom}"'] & & & & (\Gamma_{n+1} \vdash \neg G_{n+1}) \arrow[d, Rightarrow, "\widehat{\mathsf{dom}}"] \\
(\Gamma_{n+1} \vdash X_{n+1}) \arrow[rrrr, leftrightarrow, "\lesssim"] & & & & (\Gamma_{n+1} \vdash G_{n+1}) \\
& & \vdots & &
\end{tikzcd}
$$

In particular, $\Gamma_0 \vDash G_0 \gtrsim X_0$, which is Ghelli's example. Note, however, that in order to show this subtyping, we had to construct an infinite simulation: we cannot just use this lts directly in a model checker to get an algorithm for deciding subtyping of $F_{\mu\le}$. We will return to this problem in Section 5.

4 Definition of the symbolic lts semantics for $F_{\mu\le}$

We now provide formal definitions for the material discussed in Section 3. The syntax of positive labels α^+, negative labels α^- and labels α are given:

$$
\begin{aligned}
\alpha^+ &::= \tau \mid \mathsf{dom} \mid \forall X \le T \mid X \\
\alpha^- &::= \mathsf{cod} \mid \mathsf{bound} \\
\alpha &::= \alpha^+ \mid \alpha^-
\end{aligned}
$$

The symbolic lts $\xrightarrow{\alpha}$ is defined:

$$
\begin{array}{rcl}
(\Gamma \vdash T \to U) & \xrightarrow{\mathsf{dom}} & (\Gamma \vdash T) \\
(\Gamma \vdash T \to U) & \xrightarrow{\mathsf{cod}} & (\Gamma \vdash U) \\
(\Gamma \vdash K) & \xrightarrow{K} & (\Gamma \vdash \mathsf{Top}) \\
(\Gamma \vdash \forall X \le T \,.\, U) & \xrightarrow{\mathsf{bound}} & (\Gamma \vdash T) \\
(\Gamma \vdash \forall X \le T \,.\, U) & \xrightarrow{\forall X \le T} & (\Gamma, X \le T \vdash U) \\
(\Gamma \vdash X) & \xrightarrow{X} & (\Gamma \vdash \mathsf{Top}) \\
(\Gamma \vdash X) & \xrightarrow{\tau} & (\Gamma \vdash \Gamma(X)) \\
(\Gamma \vdash \mu^+ X \,.\, T) & \xrightarrow{\tau} & (\Gamma \vdash T[\mu^+ X \,.\, T / X])
\end{array}
$$

The symbolic lts $\xrightarrow{\widehat{\alpha}}$ is defined:

$$
\begin{array}{rcll}
(\Gamma \vdash T \to U) & \xrightarrow{\widehat{\mathsf{dom}}} & (\Gamma \vdash T) & \\
(\Gamma \vdash T \to U) & \xrightarrow{\widehat{\mathsf{cod}}} & (\Gamma \vdash U) & \\
(\Gamma \vdash K) & \xrightarrow{\widehat{K'}} & (\Gamma \vdash \mathsf{Top}) & (\text{when } K \le K') \\
(\Gamma \vdash \forall X \le T \,.\, U) & \xrightarrow{\widehat{\mathsf{bound}}} & (\Gamma \vdash T) & \\
(\Gamma \vdash \forall X \le T \,.\, U) & \xrightarrow{\widehat{\forall X \le V}} & (\Gamma, X \le V \vdash U) & \\
(\Gamma \vdash X) & \xrightarrow{\widehat{X}} & (\Gamma \vdash \mathsf{Top}) & \\
(\Gamma \vdash X) & \xrightarrow{\widehat{\tau}} & (\Gamma \vdash \Gamma(X)) & \\
(\Gamma \vdash \mu^+ X \,.\, T) & \xrightarrow{\widehat{\tau}} & (\Gamma \vdash T[\mu^+ X \,.\, T / X]) & \\
(\Gamma \vdash T) & \xrightarrow{\widehat{\tau}} & (\Gamma \vdash T) &
\end{array}
$$

We write $\Longrightarrow$ for the transitive reflexive closure of $\xrightarrow{\tau}$:

$$\frac{(\Gamma \vdash T) \xrightarrow{\tau} \cdots \xrightarrow{\tau} (\Gamma' \vdash T')}{(\Gamma \vdash T) \Longrightarrow (\Gamma' \vdash T')}$$

We write $\stackrel{\alpha}{\Longrightarrow}$ for the transition $\xrightarrow{\alpha}$ ignoring τ actions 'on the left', and similarly for $\stackrel{\widehat{\alpha}}{\Longrightarrow}$:

$$\frac{(\Gamma \vdash T) \Longrightarrow \cdot \xrightarrow{\alpha} (\Gamma' \vdash T')}{(\Gamma \vdash T) \stackrel{\alpha}{\Longrightarrow} (\Gamma' \vdash T')} \qquad \frac{(\Gamma \vdash T) \Longrightarrow \cdot \xrightarrow{\widehat{\alpha}} (\Gamma' \vdash T')}{(\Gamma \vdash T) \stackrel{\widehat{\alpha}}{\Longrightarrow} (\Gamma' \vdash T')}$$

A *polar simulation* $\mathcal{R}$ is a well-formed relation on types such that we can complete the diagram:

$$
\begin{tikzcd}
(\Gamma \vdash T_1) \arrow[r, leftrightarrow, "\mathcal{R}"] \arrow[d, "\alpha^\pm"'] & (\Gamma \vdash T_2) \\
(\Gamma' \vdash T_1') &
\end{tikzcd}
\quad \text{as} \quad
\begin{tikzcd}
(\Gamma \vdash T_1) \arrow[r, leftrightarrow, "\mathcal{R}"] \arrow[d, "\alpha^\pm"'] & (\Gamma \vdash T_2) \arrow[d, Rightarrow, "\widehat{\alpha^\pm}"] \\
(\Gamma' \vdash T_1') \arrow[r, leftrightarrow, "\mathcal{R}^\pm"] & (\Gamma' \vdash T_2')
\end{tikzcd}
$$

where we write $\mathcal{R}^\pm$ for:

$$\frac{(\Gamma \vdash T) \mathrel{\mathcal{R}} (\Gamma \vdash U)}{(\Gamma \vdash T) \mathrel{\mathcal{R}^+} (\Gamma \vdash U)} \qquad \frac{(\Gamma \vdash T) \mathrel{\mathcal{R}} (\Gamma \vdash U)}{(\Gamma \vdash U) \mathrel{\mathcal{R}^-} (\Gamma \vdash T)}$$

Let $\gtrsim$ be the largest polar simulation.

Proposition 2 *$\gtrsim$ is a preorder.*

Proposition 3 $\Gamma \vDash T \gtrsim U$ *iff* $\Gamma \vDash U \sqsubseteq T$.

5 Motivation for polar simulation up to polarized substitution

We have now given an alternative characterization of coinductive subtyping of $F_{\mu\le}$, but this does not directly give us any benefits. We can now use standard model-checking techniques to check subtyping, but these only terminate when they find a finite polar simulation. As the Ghelli's example (discussed in Section 3) shows, we can construct types which generate an infinite polar simulation.

In this section, we shall provide a proof technique based on Milner and Sangiorgi's [18] *bisimulation up to* methodology, which can be used to find finite representations of infinite polar simulations. It is based on the requirement to find finite symbolic graphs for process terms in Hennessy and Lin's work [11].

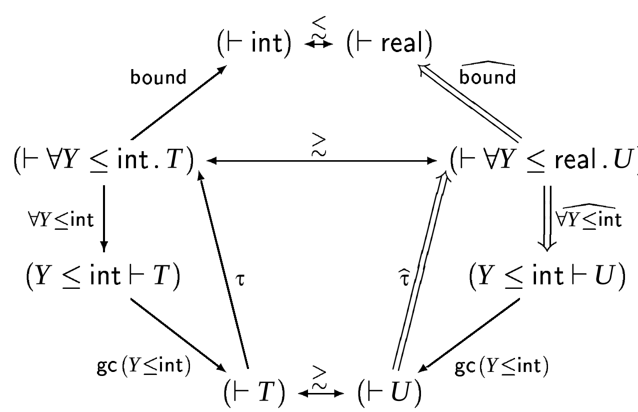
Polar simulation up to garbage collection. Define the *garbage collection* relation on well-formed types as discarding unused type variables, for example:

$$(X \le \mathsf{int}, Y \le \mathsf{real} \vdash X) \xrightarrow{\mathsf{gc}\,(Y \le \mathsf{real})} (X \le \mathsf{int} \vdash X)$$

We can use polar simulation up to garbage collection to provide finite proofs of subtyping, for example if we define:

$$T = \mu^+ X \,.\, \forall Y \le \mathsf{int} \,.\, X \qquad U = \mu^+ X \,.\, \forall Y \le \mathsf{real} \,.\, X$$

then we have a finite proof that $\vDash T \gtrsim U$ given by:



which provides us with a finite representation of the proof that $\vDash T \gtrsim U$. Polar simulation up to garbage collection is a sound proof technique, but it does not cope with Ghelli's example, since there are no unused type variables.

Polar simulation up to substitution. Our next failed attempt to find a proof technique generalizes the notion of polar simulation up to garbage collection, by observing that one can often replace a type variable by its bound, for example:

$$(X \le \mathsf{int}, Y \le X \vdash X \to Y) \xrightarrow{\mathsf{s}\,(X \le \mathsf{int})} (Y \le \mathsf{int} \vdash \mathsf{int} \to Y)$$

We can try to use this to show subtypings, for example Ghelli's $\Gamma_0 \vDash G_0 \gtrsim X_0$ from Section 3 has a finite polar simulation up to substitution:

$$
\begin{tikzcd}
(\Gamma_0 \vdash X_0) \arrow[rr, leftrightarrow, "\lesssim"] & & (\Gamma_0 \vdash \mathsf{Top}) \\
(\Gamma_0 \vdash \forall X_1 \leq X_0 \,.\, \neg X_1) \arrow[u, "\mathsf{bound}"] \arrow[rr, leftrightarrow, "\gtrsim"] \arrow[d, "\forall X_1 \leq X_0"'] & & (\Gamma_0 \vdash X_0) \arrow[u, Rightarrow, "\widehat{\mathsf{bound}}"'] \arrow[d, Rightarrow, "\widehat{\forall X_1 \leq X_0}"] \\
(\Gamma_1 \vdash \neg X_1) \arrow[d, "\mathsf{s}\,(X_1 \leq X_0)"'] & & (\Gamma_1 \vdash \neg\forall X_2 \leq X_1 \,.\, \neg X_2) \arrow[d, "\mathsf{s}\,(X_1 \leq X_0)"] \\
(\Gamma_0 \vdash \neg X_0) \arrow[rr, leftrightarrow, "\gtrsim"] \arrow[uurr, "\mathsf{dom}"'] & & (\Gamma_0 \vdash \neg\forall X_1 \leq X_0 \,.\, \neg X_1) \arrow[uull, Rightarrow, "\widehat{\mathsf{dom}}"]
\end{tikzcd}
$$

Unfortunately, polar simulation up to substitution is not a sound proof technique, for example:

$$
\begin{tikzcd}
(\vdash \forall X \leq \mathsf{int} \,.\, X) \arrow[rrr, leftrightarrow, "\not\gtrsim"] \arrow[d, "\forall X \leq \mathsf{int}"'] \arrow[dr, "\mathsf{bound}"] & & & (\vdash \forall X \leq \mathsf{int} \,.\, \mathsf{int}) \arrow[dl, Rightarrow, "\widehat{\mathsf{bound}}"'] \arrow[d, Rightarrow, "\widehat{\forall X \leq \mathsf{int}}"] \\
(X \leq \mathsf{int} \vdash X) \arrow[d, "\mathsf{s}\,(X \leq \mathsf{int})"'] & (\vdash \mathsf{int}) \arrow[r, leftrightarrow, "\lesssim"] & (\vdash \mathsf{int}) & (X \leq \mathsf{int} \vdash \mathsf{int}) \arrow[d, "\mathsf{s}\,(X \leq \mathsf{int})"] \\
(\vdash \mathsf{int}) \arrow[rrr, leftrightarrow, "\gtrsim"] & & & (\vdash \mathsf{int})
\end{tikzcd}
$$

As this example shows, we cannot always just replace type variables by their bounds, and expect to get a valid subtype relationship.

Polar simulation up to polar substitution. The technique we adopt in this paper is a refinement of polar simulation up to substitution. The crucial observation is that polar simulation up to substitution is sound, as long as we only replace negative occurrences of variables in the supertype, and positive occurrences of variables in the subtype.

Define the *positive substitution* relation as replacing any positive occurrences of a type variable by its bound, and undefined if there are any negative occurrences, for example:

$$(X \leq \mathsf{int}, Y \leq X \vdash Y \to X) \xrightarrow{\mathsf{s}^+\,(X \leq \mathsf{int})} (Y \leq \mathsf{int} \vdash Y \to \mathsf{int})$$
$$(X \leq \mathsf{int}, Y \leq X \vdash X \to Y) \;\not\xrightarrow{\mathsf{s}^+\,(X \leq \mathsf{int})}\; (Y \leq \mathsf{int} \vdash \mathsf{int} \to Y)$$

and the *negative substitution* relation similarly (but note that we always substitute positively in the type context):

$$(X \leq \mathsf{int}, Y \leq X \vdash X \to Y) \xrightarrow{\mathsf{s}^-\,(X \leq \mathsf{int})} (Y \leq \mathsf{int} \vdash \mathsf{int} \to Y)$$

Then a *polar simulation up to polar substitution* is one where we are allowed to use negative substitution in the supertype, and positive substitution in the subtype. For example, we now have a valid finite proof of Ghelli's example:

$$
\begin{tikzcd}
(\Gamma_0 \vdash X_0) \arrow[rr, leftrightarrow, "\lesssim"] & & (\Gamma_0 \vdash \mathsf{Top}) \\
(\Gamma_0 \vdash \forall X_1 \leq X_0 \,.\, \neg X_1) \arrow[u, "\mathsf{bound}"] \arrow[rr, leftrightarrow, "\gtrsim"] \arrow[d, "\forall X_1 \leq X_0"'] & & (\Gamma_0 \vdash X_0) \arrow[u, Rightarrow, "\widehat{\mathsf{bound}}"'] \arrow[d, Rightarrow, "\widehat{\forall X_1 \leq X_0}"] \\
(\Gamma_1 \vdash \neg X_1) \arrow[d, "\mathsf{s}^-\,(X_1 \leq X_0)"'] & & (\Gamma_1 \vdash \neg\forall X_2 \leq X_1 \,.\, \neg X_2) \arrow[d, "\mathsf{s}^+\,(X_1 \leq X_0)"] \\
(\Gamma_0 \vdash \neg X_0) \arrow[rr, leftrightarrow, "\gtrsim"] \arrow[uurr, "\mathsf{dom}"'] & & (\Gamma_0 \vdash \neg\forall X_1 \leq X_0 \,.\, \neg X_1) \arrow[uull, Rightarrow, "\widehat{\mathsf{dom}}"]
\end{tikzcd}
$$

and the counterexample for polar simulation up to substitution is no longer a counterexample, because it does not use substitution with the right polarity.

Polar simulation up to polar substitution is the proof technique we adopt for the rest of this paper.

6 Definition of polar simulation up to polar substitution

Let the *garbage collection relation* $(\Gamma \vdash T) \xrightarrow{\mathsf{gc}\,\Delta} (\Gamma' \vdash T')$ be:

$$(\Gamma, \Delta \vdash T) \xrightarrow{\mathsf{gc}\,\Delta} (\Gamma \vdash T) \quad \text{(when } \Gamma \vdash T\text{)}$$

Let $\mathcal{R}$ be a polar simulation *up to garbage collection* whenever we can complete any diagram:

$$
\begin{tikzcd}
(\Gamma \vdash T_1) \arrow[r, leftrightarrow, "\mathcal{R}"] \arrow[d, "\alpha^{\pm}"'] & (\Gamma \vdash T_2) \\
(\Gamma' \vdash T_1') &
\end{tikzcd}
\quad \text{as} \quad
\begin{tikzcd}
(\Gamma \vdash T_1) \arrow[r, leftrightarrow, "\mathcal{R}"] \arrow[d, "\alpha^{\pm}"'] & (\Gamma \vdash T_2) \arrow[d, Rightarrow, "\widehat{\alpha^{\pm}}"'] \\
(\Gamma' \vdash T_1') \arrow[d, "\mathsf{gc}\,\Delta"'] & (\Gamma' \vdash T_2') \arrow[d, "\mathsf{gc}\,\Delta"] \\
(\Gamma'' \vdash T_1'') \arrow[r, leftrightarrow, "\mathcal{R}^{\pm}"] & (\Gamma'' \vdash T_2'')
\end{tikzcd}
$$

Define a *polar substitution* $T[U/X]^{\pm}$ as:

$$T[U/X]^{\pm} \quad = \quad T[U/X] \quad \text{(when } X \notin \mathsf{fv}^{\mp}(T)\text{)}$$

Define a *polar context substitution* $T[\Delta]^{\pm}$ as:

$$
\begin{aligned}
T[\emptyset]^{\pm} &\quad = \quad T \\
T[\Delta, X \leq U]^{\pm} &\quad = \quad T[U/X]^{\pm}[\Delta]^{\pm} \quad \text{(when } X \notin \mathsf{fv}(\Delta)\text{)}
\end{aligned}
$$

Define a *polar substitution relation* $(\Gamma \vdash T) \xrightarrow{\mathsf{s}^{\pm}\,\Delta} (\Gamma' \vdash T')$ as:

$$(\Gamma, \Delta \vdash T) \xrightarrow{\mathsf{s}^{\pm}\,\Delta} (\Gamma[\Delta]^{+} \vdash T[\Delta]^{\pm})$$

Note that polar substitution generalizes garbage collection:

$$\text{if } (\Gamma \vdash T) \xrightarrow{\mathsf{gc}\,\Delta} (\Gamma' \vdash T') \text{ then } (\Gamma \vdash T) \xrightarrow{\mathsf{s}^{\pm}\,\Delta} (\Gamma' \vdash T')$$

Let $\mathcal{R}$ be a polar simulation *up to polar substitution* whenever we can complete any diagram:

$$\begin{tikzcd} (\Gamma \vdash T_1) \arrow[r, leftrightarrow, "\mathcal{R}"] \arrow[d, "\alpha^{\pm}"'] & (\Gamma \vdash T_2) \\ (\Gamma' \vdash T_1') & \end{tikzcd} \quad \text{as} \quad \begin{tikzcd} (\Gamma \vdash T_1) \arrow[r, leftrightarrow, "\mathcal{R}"] \arrow[d, "\alpha^{\pm}"'] & (\Gamma \vdash T_2) \arrow[d, Rightarrow, "\widehat{\alpha^{\pm}}"] \\ (\Gamma' \vdash T_1') \arrow[d, "\mathsf{s}^{\mp}\,\Delta"'] & (\Gamma' \vdash T_2') \arrow[d, "\mathsf{s}^{\pm}\,\Delta"] \\ (\Gamma'' \vdash T_1'') \arrow[r, leftrightarrow, "\mathcal{R}^{\pm}"] & (\Gamma'' \vdash T_2'') \end{tikzcd}$$

We can then show that polar simulation up to polar substitution (and hence up to garbage collection) is a sound proof technique.

Proposition 4 *If $\mathcal{R}$ is a polar simulation up to polar substitution and $\Gamma \vDash T \mathrel{\mathcal{R}} U$ then $\Gamma \vDash T \gtrsim U$.*

7 An algorithm for finding polar simulation up to polar substitution

Polar simulation up to polar substitution gives us a proof technique for showing subtyping, which can easily be converted into a model checking algorithm. Since $F_{\mu\leq}$ is deterministic, a simple breadth-first search algorithm is sufficient. The algorithm is given in Figure 1. The invariants for the **while** loop in the algorithm are:

1. Either $\Gamma_0 \vDash T_0 \mathrel{\mathcal{R}} U_0$ or $\Gamma_0 \vDash T_0 \mathrel{\mathcal{S}} U_0$.
2. $\mathcal{R}$ is a polar simulation up to polar substitution mod $\mathcal{S}$.
3. If $\Gamma_0 \vDash T_0 \gtrsim U_0$ then $(\mathcal{R} \cup \mathcal{S}) \subseteq \gtrsim$.

where $\mathcal{R}$ is a polar simulation up to polar substitution *mod* $\mathcal{S}$ whenever we can complete any diagram:

$$\begin{tikzcd} (\Gamma \vdash T_1) \arrow[r, leftrightarrow, "\mathcal{R}"] \arrow[d, "\alpha^{\pm}"'] & (\Gamma \vdash T_2) \\ (\Gamma' \vdash T_1') & \end{tikzcd} \quad \text{as} \quad \begin{tikzcd} (\Gamma \vdash T_1) \arrow[r, leftrightarrow, "\mathcal{R}"] \arrow[d, "\alpha^{\pm}"'] & (\Gamma \vdash T_2) \arrow[d, Rightarrow, "\widehat{\alpha^{\pm}}"'] \\ (\Gamma' \vdash T_1') \arrow[d, "\mathsf{s}^{\mp}\,\Delta"'] & (\Gamma' \vdash T_2') \arrow[d, "\mathsf{s}^{\pm}\,\Delta"] \\ (\Gamma'' \vdash T_1'') \arrow[r, leftrightarrow, "\mathcal{R}^{\pm} \cup \mathcal{S}^{\pm}"] & (\Gamma'' \vdash T_2'') \end{tikzcd}$$

It is not too difficult to establish partial correctness of this algorithm, by establishing Invariants 1–3:

function *suptype* (Γ_0, T_0, U_0) $\{$
$\quad$**let** $\mathcal{R} = \emptyset$;
$\quad$**let** $\mathcal{S} = \{\Gamma_0 \vDash T_0 \mathrel{\mathcal{S}} U_0\}$;
$\quad$**while** $(\mathcal{S} \neq \emptyset)$ $\{$
$\quad\quad$**let** $\mathcal{S}' = \emptyset$;
$\quad\quad$**foreach** $(\Gamma_1 \vDash T_1 \mathrel{\mathcal{S}} U_1)$ $\{$
$\quad\quad\quad$**foreach** $(\Gamma_1 \vdash T_1) \xrightarrow{\alpha^{\pm}} (\Gamma_2 \vdash T_2)$ $\{$
$\quad\quad\quad\quad$**if** $(\alpha^{\pm} = \tau)$ $\{$
$\quad\quad\quad\quad\quad$**add** $\Gamma_2 \vDash T_2 \mathrel{\mathcal{S}'} U_1$ **to** $\mathcal{S}'$;
$\quad\quad\quad\quad$$\}$ **else if** $(\Gamma_1 \vdash U_1) \stackrel{\widehat{\alpha^{\pm}}}{\Longrightarrow} (\Gamma_2 \vdash U_2)$ $\{$
$\quad\quad\quad\quad\quad$**let** Δ be the largest type context
$\quad\quad\quad\quad\quad$such that $(\Gamma_2 \vdash T_2) \xrightarrow{\mathsf{s}^{\mp}\,\Delta} (\Gamma_3 \vdash T_3)$
$\quad\quad\quad\quad\quad$and $(\Gamma_2 \vdash U_2) \xrightarrow{\mathsf{s}^{\pm}\,\Delta} (\Gamma_3 \vdash U_3)$;
$\quad\quad\quad\quad\quad$**add** $\Gamma_3 \vDash T_3 \mathrel{\mathcal{S}'^{\pm}} U_3$ **to** $\mathcal{S}'^{\pm}$;
$\quad\quad\quad\quad$$\}$ **else** $\{$
$\quad\quad\quad\quad\quad$**return** *false*;
$\quad\quad\quad\quad$$\}$
$\quad\quad\quad$$\}$
$\quad\quad$$\}$
$\quad\quad$$\mathcal{R} = \mathcal{R} \cup \mathcal{S}$;
$\quad\quad$$\mathcal{S} = \mathcal{S}' \setminus \mathcal{R}$;
$\quad$$\}$
$\quad$**return** *true*;
$\}$

Figure 1: The algorithm

Proposition 5 *For any $\Gamma_0 \vdash T_0$ and $\Gamma_0 \vdash U_0$ we have:*

1. *If suptype (Γ_0, T_0, U_0) returns true then $\Gamma_0 \vDash T_0 \gtrsim U_0$.*
2. *If suptype (Γ_0, T_0, U_0) returns false then $\Gamma_0 \vDash T_0 \not\gtrsim U_0$.*

We can show that the algorithm is guaranteed to terminate in the case where $\Gamma \vDash T \not\gtrsim U$.

Proposition 6 *If $\Gamma \vDash T \not\gtrsim U$ then suptype (Γ, T, U) terminates.*

We can also show that if there is a finite polar simulation up to polar substitution, then the algorithm will find it, and so will terminate. For example, this means the algorithm is guaranteed to terminate on Ghelli's example.

Proposition 7 *If there exists a finite polar simulation up to polar substitution $\mathcal{R}_f$ such that $\Gamma \vDash T \mathrel{\mathcal{R}_f} U$ then suptype (Γ, T, U) terminates.*

Using this, we can show that the algorithm is at least as strong as the standard algorithm for subtyping $F_{\leq}$. We do this by showing that if $\Gamma \vdash T \geq U$ then we can construct a finite polar simulation $\mathcal{R}$ such that $\Gamma \vdash T \mathrel{\mathcal{R}} U$.

Proposition 8 *If the standard algorithm for subtyping $F_{\leq}$ terminates, then suptype (Γ, T, U) terminates with the same result.*

Since our algorithm is at least as powerful as the standard algorithm, but terminates on Ghelli's example, we have that our example is strictly more powerful.

8 Kernel $F_{\mu\leq}$

In [4], Colazzo and Ghelli provide an algorithm for subtyping of Kernel $F_{\mu\leq}$. Their algorithm:

- Works directly on the structure of the types, rather than via an lts semantics.
- Does not work 'up to α-conversion', which results in a more efficient algorithm, at the cost of extra complexity.

We can easily modify our algorithm to check Kernel $F_{\mu\leq}$ subtyping, by changing the matching transition rule for polymorphic types to require bounds to be matched exactly:

$$(\Gamma \vdash \forall X \leq T \,.\, U) \xrightarrow{\widehat{\forall X \leq T}} (\Gamma, X \leq T \vdash U)$$

We can show that this modified algorithm is as powerful as theirs (although probably not as efficient, depending on how α-conversion is handled), by showing that our algorithm terminates on Kernel $F_{\mu\leq}$.

Proposition 9 *If $\Gamma \vDash T \gtrsim U$ in Kernel $F_{\mu\leq}$, then there is a finite polar simulation $\mathcal{R}$ up to garbage collection such that $\Gamma \vDash T \mathrel{\mathcal{R}} U$.*

Together with Proposition 7, this gives us that our algorithm is a decision procedure for subtyping of Kernel $F_{\mu\leq}$.

Proposition 10 *If $\Gamma \vDash T \gtrsim U$ in Kernel $F_{\mu\leq}$, then suptype (Γ, T, U) terminates with true.*

9 Colazzo and Ghelli's benchmark examples

We have already shown that our algorithm terminates on Ghelli's example of nontermination of the standard subtyping algorithm for $F_{\leq}$.

Colazzo and Ghelli [4] provide two motivating examples for their algorithm for Kernel $F_{\mu\leq}$, which act as useful benchmarks for our approach. The examples make use of tuple types $T \times U$, and a bottom type $\bot$: these can easily be given an lts semantics:

$$\begin{tikzcd}
 & (\Gamma \vdash T \times U) \arrow[dl, "\mathsf{fst}"'] \arrow[dr, "\mathsf{snd}"] & & (\Gamma \vdash \bot) \arrow[d, "\bot"'] \\
(\Gamma \vdash T) & & (\Gamma \vdash U) & (\Gamma \vdash \bot)
\end{tikzcd}$$

with matching transitions:

$$\begin{tikzcd}
 & (\Gamma \vdash T \times U) \arrow[dl, "\widehat{\mathsf{fst}}"'] \arrow[dr, "\widehat{\mathsf{snd}}"] & & & (\Gamma \vdash \bot) \arrow[dl, "\widehat{\alpha^-}"'] \arrow[dr, "\widehat{\alpha^+}"] & \\
(\Gamma \vdash T) & & (\Gamma \vdash U) & (\Gamma \vdash \mathsf{Top}) & & (\Gamma \vdash \bot)
\end{tikzcd}$$

For example, we can use this semantics to verify one of Pierce's [15] requirements for subtyping with $\bot$, that any type variable bounded by $\bot$ is equivalent to $\bot$:

$$X \leq \bot \vDash X \gtrsim \bot \qquad X \leq \bot \vDash \bot \gtrsim X$$

In the examples, we also use many syntactic abbreviations, such as defining equations, missing Top bounds, and ignoring some τ steps.

The first example is a benchmark which checks that the algorithm performs enough garbage collection to find a finite polar simulation up to garbage collection. It is given in Figure 2.

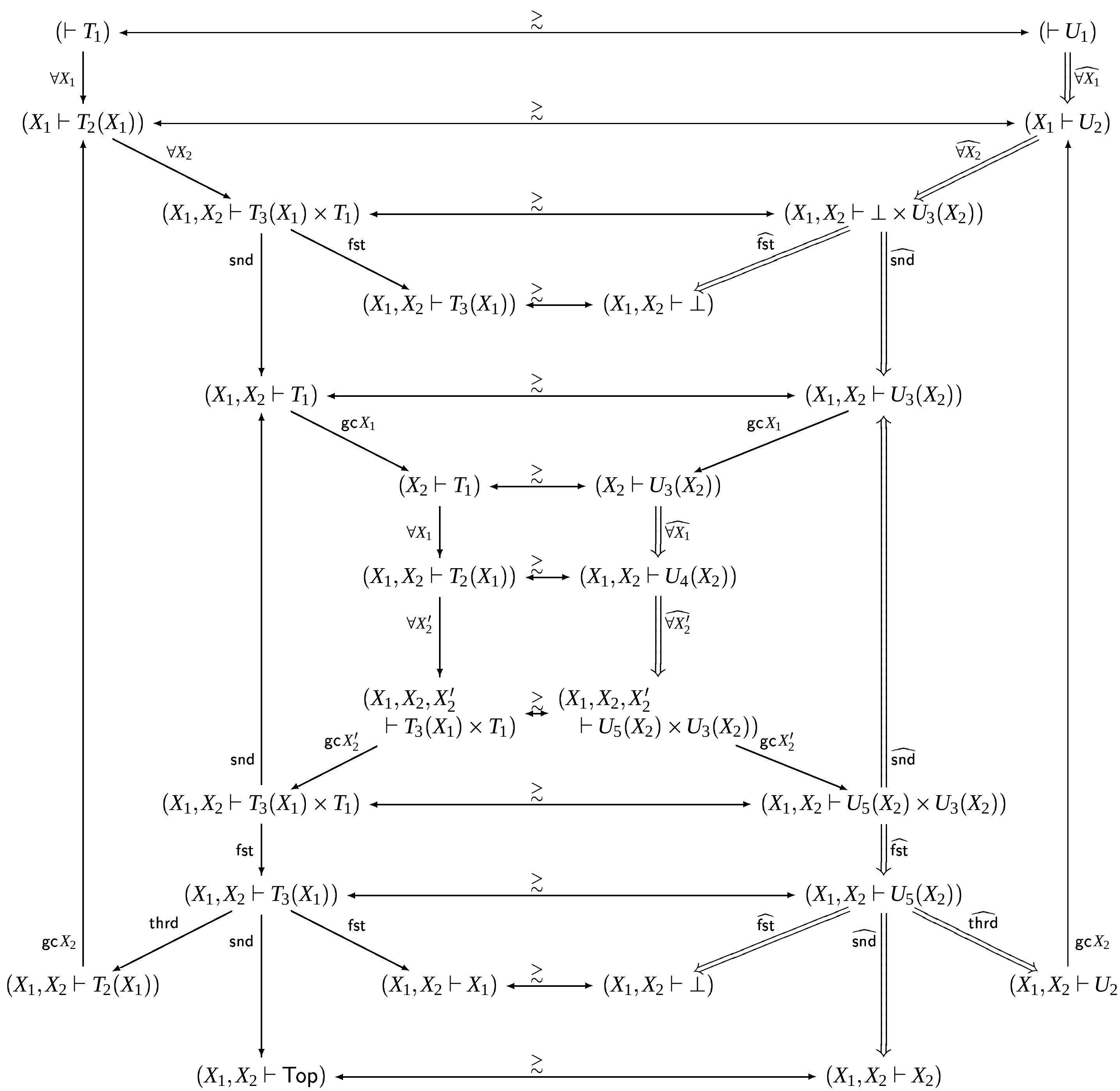
The second example checks that the algorithm does not produce false positives, caused by collapsing variables together incorrectly. It is given in Figure 3.

10 Conclusions and further work

This paper describes an application of symbolic labelled transition systems, which have previously been used to model concurrent languages, to modelling subtyping. This allows us to use the techniques from concurrency theory, such as simulations, and 'simulation up to' to reason about subtyping. It also often makes proofs easier to read, even in the presence of quite complex types such as Colazzo and Ghelli's benchmark in Figure 2.

This technique should generalize to other examples such as record subtyping, union types and intersection types. It may be that Gordon's [10] work on lts semantics for λ-calculi could be applied here, to give a semantics of higher-order features such as functions of kind $\mathsf{Type} \to \mathsf{Type}$. We leave the technical development of this to future work.

The main result which is missing from the current work is a syntactic characterization of when the algorithm *suptype* terminates. Also, we have not discussed how α-conversion would be implemented: it should be possible to define α-conversion as a strong bisimulation, and then use polar simulation up to strong bisimulation as a proof technique. We also leave these issues for future work.



$$
\begin{array}{rcl}
T_1 & \stackrel{\text{def}}{=} & \forall X_1 \,.\, T_2(X_1) \\
T_2(X_1) & \stackrel{\text{def}}{=} & \forall X_2 \,.\, (T_3(X_1) \times T_1) \\
T_3(X_1) & \stackrel{\text{def}}{=} & X_1 \times \mathsf{Top} \times T_2(X_1)
\end{array}
\qquad
\begin{array}{rcl}
U_1 & \stackrel{\text{def}}{=} & \forall Y_1 \,.\, U_2 \\
U_2 & \stackrel{\text{def}}{=} & \forall Y_2 \,.\, (\bot \times U_3(Y_2)) \\
U_3(Y_2) & \stackrel{\text{def}}{=} & \forall Y_3 \,.\, U_4(Y_2) \\
U_4(Y_2) & \stackrel{\text{def}}{=} & \forall Y_4 \,.\, (U_5(Y_2) \times U_3(Y_2)) \\
U_5(Y_2) & \stackrel{\text{def}}{=} & \bot \times Y_2 \times U_2
\end{array}
$$

Figure 2: Colazzo and Ghelli's first example: show that $\vdash T_1 \gtrsim U_1$

$$
\begin{tikzcd}
(\vdash T_1) \arrow[r, leftrightarrow, "\not\gtrsim"] \arrow[d, "\forall X_1"'] & (\vdash U_1) \arrow[d, Rightarrow, "\widehat{\forall X_1}"'] \\
(X_1 \vdash T_2(X_1)) \arrow[r, leftrightarrow, "\not\gtrsim"] \arrow[d, "\mathsf{fst}"'] & (X_1 \vdash U_2(X_1)) \arrow[d, Rightarrow, "\widehat{\mathsf{fst}}"'] \\
(X_1 \vdash T_3(X_1)) \arrow[r, leftrightarrow, "\not\gtrsim"] \arrow[d, "\mathsf{snd}"'] & (X_1 \vdash U_2(X_1)) \arrow[d, Rightarrow, "\widehat{\mathsf{snd}}"'] \\
(X_1 \vdash T_4(X_1)) \arrow[r, leftrightarrow, "\not\gtrsim"] \arrow[d, "\mathsf{snd}"'] & (X_1 \vdash U_3(X_1)) \arrow[d, Rightarrow, "\widehat{\mathsf{snd}}"'] \\
(X_1 \vdash T_5(X_1)) \arrow[r, leftrightarrow, "\not\gtrsim"] \arrow[d, "\forall X_5"'] & (X_1 \vdash U_1) \arrow[d, Rightarrow, "\widehat{\forall X_5}"'] \\
(X_1, X_5 \vdash T_2(X_1)) \arrow[r, leftrightarrow, "\not\gtrsim"] \arrow[d, "\mathsf{fst}"'] & (X_1, X_5 \vdash U_2(X_5)) \arrow[d, Rightarrow, "\widehat{\mathsf{fst}}"'] \\
(X_1, X_5 \vdash T_3(X_1)) \arrow[r, leftrightarrow, "\not\gtrsim"] \arrow[d, "\mathsf{snd}"'] & (X_1, X_5 \vdash U_2(X_5)) \arrow[d, Rightarrow, "\widehat{\mathsf{snd}}"'] \\
(X_1, X_5 \vdash T_4(X_1)) \arrow[r, leftrightarrow, "\not\gtrsim"] \arrow[d, "\mathsf{fst}"'] & (X_1, X_5 \vdash U_3(X_5)) \arrow[d, Rightarrow, "\widehat{\mathsf{fst}}"'] \\
(X_1, X_5 \vdash X_1) \arrow[r, leftrightarrow, "\not\gtrsim"] \arrow[d, "X_1"'] & (X_1, X_5 \vdash X_5) \\
(X_1, X_5 \vdash \mathsf{Top}) &
\end{tikzcd}
$$

$$
\begin{aligned}
T_1 &\stackrel{\mathsf{def}}{=} \forall X_1 \,.\, T_2(X_1) \\
T_2(X_1) &\stackrel{\mathsf{def}}{=} T_3(X_1) \times \mathsf{Top} \\
T_3(X_1) &\stackrel{\mathsf{def}}{=} \mathsf{Top} \times T_4(X_1) \\
T_4(X_1) &\stackrel{\mathsf{def}}{=} X_1 \times T_5(X_1) \\
T_5(X_1) &\stackrel{\mathsf{def}}{=} \forall X_5 \,.\, T_2(X_1)
\end{aligned}
$$

$$
\begin{aligned}
U_1 &\stackrel{\mathsf{def}}{=} \forall Y_1 \,.\, U_2(Y_1) \\
U_2(Y_1) &\stackrel{\mathsf{def}}{=} U_2(Y_1) \times U_3(Y_1) \\
U_3(Y_1) &\stackrel{\mathsf{def}}{=} Y_1 \times U_1
\end{aligned}
$$

Figure 3: Colazzo and Ghelli's second example: show $\vDash T_1 \not\gtrsim U_1$

References

[1] M. Brandt and F. Henglein. Coinductive axiomatization of recursive type equality and subbtyping. In *Proc. Typed Lambda Calculi and Applications*, volume 1210 of *Lecture Notes in Computer Science*, pages 63–81. Springer-Verlag, 1997.

[2] K. B. Bruce and G. Longo. A modest model of records, inheritance and bounded quantification. *Inform. and Comput.*, 87(1):196–240, 1990.

[3] L. Cardelli and P. Wegner. On understanding types, data abstraction and polymorphism. *Computing Surveys*, 17(4):471–522, 1985.

[4] D. Colazzo and G. Ghelli. Subtyping recursive types in kernel Fun, extended abstract. In *Proc. Logic in Computer Science*. IEEE Computer Society Press, 1999.

[5] P.-L. Curien and G. Ghelli. Coherence of subsumtion: Minimum typing and type checking in $F_{\leq}$. *Math. Struct. in Comp. Sci.*, 2(1):55–91, 1992.

[6] V. Gapeyev, M. Levin, and B. C. Pierce. Recursive subtyping revealed. In *Proc. Int. Conf. Functional Programming*, 2000.

[7] G. Ghelli. *Proof Theoretic Studies about a Minimal Type System Integrating Inclusion and Parametric Polymorphism*. PhD thesis, Universitá di Pisa, 1990.

[8] G. Ghelli. Recursive types are not conservative over $F_{\leq}$. In M. Bezen and J.F. Groote, editors, *Proc. Typed Lambda Calculi and Applications*, number 664 in Lecture Notes in Computer Science, pages 146–162. Springer-Verlag, 1993.

[9] G. Ghelli. Divergence of $\leq$ type checking. *Theoret. Comput. Sci.*, 139(1-2):131–162, 1995.

[10] A. D. Gordon. Bisimilarity as a theory of functional programming. In *Proc. Math. Foundations of Programming Semantics*, number 1 in Electronic Notes in Comp. Sci. Elsevier, 1995.

[11] M. Hennessy and H. Lin. Symbolic bisimulations. *Theoret. Comput. Sci.*, pages 353–389, 1995.

[12] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[13] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[14] B. C. Pierce. Bounded quantification is undecidable. *Inform. and Comput.*, 112(1):131–165, 1994.

[15] B. C. Pierce. Bounded quantification with bottom. Technical Report 492, Computer Science Department, Indiana University, 1997.

[16] B.C. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. In *Proc. LICS '93*, pages 376–385. IEEE Computer Society Press, 1993.

[17] M. Amadio R and L. Cardelli. Subtyping recursive types. *ACM Trans. Programming Languages and Systems*, 15(4):575–631, 1993.

[18] D. Sangiorgi and R. Milner. The problem of 'weak bisimulation up to'. In *Proc. CONCUR 92*, volume 630 of *Lecture Notes in Computer Science*. Springer Verlag, 1992.

[19] P. Sewell. Global/local subtyping for a distributed π-calculus. Technical Report 435, Computer Laboratory, University of Cambridge, 1997.

[20] D. N. Turner. *The Polymorphic Pi-calculus: Theory and Implementation*. PhD thesis, University of Edinburgh, 1995.