Binary session types for psi-calculi

Hans Hüttel

Department of Computer Science, Aalborg University, Denmark

1 Introduction

Binary session types arose in the work of Honda in the setting of the π-calculus; a binary session type describes the protocol followed by the two ends of a communication channel and a well-typed process will not exhibit communication errors, since the two ends of a channel must always adhere to the dual parts of their protocol. Binary session types have been used for describing a variety of program properties, including liveness properties. In the setting of sessions, an important property is that of progress , namely that a session will never be stuck waiting for a message that does not arrive.

This paper provides a common generalization of existing binary session type systems using psi-calculi . These have been proposed as a common framework for understanding the plethora of π-like process calculi; like π-like calculi psi-calculi have the notion of mobile names with scope but also allow channels to be not just names but arbitrary terms from a so-called nominal data type.

Type systems for psi-calculi already exist. In particular, there is a type system generalizing a collection of simple type systems and another system for resource-aware properties . We now extend this approach to a generic type system for binary session types. We only assume that session types have certain labelled transitions; this is in the tradition of behavioural contracts that provides a behavioural type discipline in which types have transitions.

A main result is the definition of a binary session type system for psi-calculi and a fidelity result that generalizes results from existing session type systems. Since channels can be arbitrary terms, a major challenge is to deal with this. Whenever a session is created, private session channels are introduced by means of scoped endpoint constructors that can be applied to ordinary terms in order to create a session channel. The type system keeps track of how the behaviour of a session channel evolves by keeping track of the modified behaviour of these endpoint constructors.

The safety result for our binary session type discipline is that of fidelity, namely that the usage of a well-typed channel evolves according to its session type.

Existing binary session type systems arise as instances of our general type system. These include a system for ensuring progress due to Vieira and Vasconcelos, a type system for correspondence assertions due to Vasconcelos et al. and the system with subtyping due to Gay and Hole .

2 Psi-calculi

Psi-calculus processes can contain *terms* $M, N, \ldots$; these must form a nominal datatype $\mathbf{T}$. If Σ is a signature, a nominal data type is then a Σ-algebra, whose carrier set is a nominal set. In the nominal data types of ψ-calculi we use simultaneous term substitution $X[\widetilde{z} := \widetilde{Y}]$ which is to be read as stating that the terms in $\widetilde{Y}$ replace the names in $\widetilde{z}$ in X. We assume a notion of channel equivalence; $\Psi \models M \leftrightarrows N$ denotes that terms M and N represent the same channel.

Processes can also contain *assertions* Ψ and *conditions* φ, that must also form nominal datatypes.

Unlike in the π-calculus channels can be arbitrary terms in a psi-calculus, as arbitrary terms are allowed in the subject position of a prefix.

We extend psi-calculus with the selection and branching primitives of Honda et al. , as these are standard in session calculi. In a selector was always a name; here we allow arbitrary terms M as selectors. Branching thus becomes $M \triangleright \{l_1 : P_1, \ldots, l_: P_k\}$ and selection is written as $M \triangleleft l.P_1$, where l ranges over a set of label names.

We introduce session channels by means of dual endpoints as in Giunti and Vasconcelos . The construct $(\nu c)P$ can be used to set up a new session channel with endpoint constructor c that can be used to build session channels.

All in all, this gives us the formation rules

$$\begin{aligned} P ::= & \underline{M}(\lambda \widetilde{x})X.P \mid \overline{M}N.P \mid P_1 \mid P_2 \mid (\nu c)P \mid !\,P \mid (\!|\Psi|\!) \mid \mathbf{case}\ \varphi_1 : P_1, \ldots, \varphi_k : P_k \\ & \mid M \triangleleft l.P_1 \mid M \triangleright \{l_1 : P_1, \ldots, l_k : P_k\} \end{aligned}$$

3 A generic type system

We T range over the set of types and distinguish between *base types* B, *session types* S and *endpoint types* T_E. An endpoint type T_E describes the behaviour at one end of a channel. A session type S describes the behaviour at both ends of a channel and is an unordered pair (T_1, T_2) of endpoint types, i.e. so (T_1, T_2) and (T_2, T_1) denote the same type.

In psi-calculi channels can be arbitrary terms; in our setting we use *session constructors* to indicate that a term is to be used as a session channel. A term whose principal session constructor is c will have a type of the form $T@c$.

We assume a deterministic labelled transition relation defined on the set of endpoint types. Transitions are of the form $T_E \xrightarrow{\lambda} T'$ where

$$\lambda ::= !T_1 \mid ?T_1 \mid \triangleleft l \mid \triangleright l$$

If a channel has endpoint type T_E, which has the transition $T_E \xrightarrow{?T_1} T'_E$, then following an input of a term of type T_1, the channel will now have endpoint type T'_E. For a given type language, we must give transition rules that describe how these transitions arise.

We assume a *duality condition* for labels in labelled type transitions; we define $\overline{!T_1} = ?T_2$ and $\overline{\triangleleft l} = \triangleright l$ and vice versa, and we require that $\overline{\overline{\lambda}} = \lambda$. A session type is balanced if the types of its endpoint are dual to each other.

Definition 1. A session type S is balanced if $S = (T_E, \overline{T_E})$ for some T_E.

A type environment Γ is a finite function from names to types, often written as $\widetilde{x} : \widetilde{T}$. A type environment Γ is balanced if for every $x \in \mathrm{dom}(\Gamma)$ we have that whenever $\Gamma(x) = S$, then S is balanced.

Type judgements The type judgements in our type system are of the form $\Gamma, \Psi \vdash \mathcal{J}$ where $\mathcal{J}$ is built using the formation rules

$$\mathcal{J} ::= M : T \mid X : \widetilde{T} \to U \mid \Psi \mid \varphi \mid P$$

For terms, the type judgment $\Gamma, \Psi \vdash M : T@c$ says that the term M has type T using session constructor c. The rules defining these judgements depend on the instance of the type system but we require that the session constructor must have an endpoint type for the resulting channel to be typable. Rules for assertions and conditions are also specific to the instance considered.

The type rules for *processes* contain judgment of the form $\Gamma, \Psi \vdash P$ where Ψ is an assertion. Table 1 contains the most interesting type rules for processes.

Note that for patterns, judgments are of the form $\Gamma, \Psi \vdash X : \widetilde{T} \to U$. The intended interpretation is that pattern X has type $\widetilde{T} \to U$ if the pattern variables are bound to terms of types $\widetilde{T}$ whenever the pattern matches a term of type U.

An important rule is that for parallel composition, since type addition enables us to split a session type into two endpoint types; this follows Giunti and Vasconcelos .

Whenever a prefix is typed, the type of the subject must be updated when typing the continuation. As subjects in the psi-calculus setting can be arbitrary terms, we update the type of the channel constructor used to construct the channel.

$$\text{(Output)} \quad \frac{\Gamma_1, \Psi_1 \vdash_{\min} M : T_1@c \qquad T_1 \xrightarrow{!,T_2} T_3 \qquad \Gamma_2, \Psi_2 \vdash_{\min} N : T_2 \qquad \Gamma_3 + c : T_3, \Psi_3 \vdash P}{\Gamma_1 + \Gamma_2 + \Gamma_3, \Psi_1 \otimes \Psi_2 \otimes \Psi_3 \vdash \overline{M}N.P}$$

$$\text{(Input)} \quad \frac{\Gamma_1, \Psi_1 \vdash_{\min} M : T_1@c \qquad T_1 \xrightarrow{?,T_2} T_3(\widetilde{x}) \qquad \Gamma_2, \Psi_2 \vdash_{\min} X : \widetilde{U} \to T_2 \qquad \Gamma_3 + \widetilde{x} : \widetilde{U} + c : T_3[\widetilde{x}], \Psi_3 \vdash P}{\Gamma_1 + \Gamma_2 + \Gamma_3, \Psi_1 \otimes \Psi_2 \otimes \Psi_3 \vdash \underline{M}(\lambda \widetilde{x})X.P} \quad \begin{array}{l} \widetilde{x} \sharp \operatorname{dom}(\Gamma_1 + \Gamma_2 + \Gamma_3) \\ \widetilde{x} \sharp \Psi_1 \otimes \Psi_2 \otimes \Psi_3 \end{array}$$

$$\text{(Case)} \quad \frac{\Gamma, \Psi \vdash \varphi_i \quad \Gamma, \Psi \vdash P_i \quad 1 \leq i \leq k}{\Gamma, \Psi \vdash \mathbf{case}\ \varphi_1 : P_1, \ldots, \varphi_k : P_k} \qquad \text{(Session)} \quad \frac{\Gamma + x : T, \Psi \vdash P}{\Gamma, \Psi \vdash (\nu x : T)P} \quad x \sharp \Gamma, \Psi$$

$$\text{(Select)} \quad \frac{\Gamma_1, \Psi_1 \vdash_{\min} M : T@c \qquad \Gamma_2 + c : T_i, \Psi_2 \vdash P \qquad T \xrightarrow{\lhd, l_i} T_i}{\Gamma_1 + \Gamma_2, \Psi_1 \otimes \Psi_2 \vdash M \lhd l_i.P}$$

$$\text{(Branch)} \quad \frac{\Gamma_1, \Psi_1 \vdash_{\min} M : T@c \qquad T \xrightarrow{\rhd, l_i} T_i \text{ and } \Gamma_2 + c : T_i, \Psi_{2i} \vdash P_i \text{ for } 1 \leq i \leq k}{\Gamma_1 + \Gamma_2, \Psi_1 \otimes \bigotimes_{i=1}^{k} \Psi_{2i} \vdash M \rhd \{l_1 : P_1, \ldots, l_k : P_k\}}$$

Table 1: Selected type rules

Theorem 1. *Suppose we have $\Psi_0 \blacktriangleright P \xrightarrow{\alpha} P'$, where α is a τ-action and that $\Gamma, \Psi \vdash_{\mathsf{bal}} P$ and $\Psi \leq \Psi_0$. Then for some $\Psi' \leq \Psi$ and $\Gamma' \leq \Gamma$ we have $\Gamma' \vdash_{\min} \alpha : (T@c, U)$.*

Theorem 2 (Fidelity)**.** *Suppose we have $\Psi_0 \blacktriangleright P \xrightarrow{\alpha} P'$, where α is a τ-action and that $\Gamma, \Psi \vdash P$ with Γ and Γ_P balanced and $\Psi \leq \Psi_0$. Then for some $\Psi' \leq \Psi$ we have $\Gamma \uparrow \alpha, \Psi' \vdash_{\mathsf{bal}} P'$.*