Polymorphic Subtyping for Effect Analysis: the Dynamic Semantics

Torben Amtoft & Flemming Nielson & Hanne Riis Nielson & Jürgen Ammann

Computer Science Department, Aarhus University, Denmark

e-mail: {tamtoft,fnielson,hrnielson}@daimi.aau.dk

Abstract. We study an annotated type and effect system that integrates `let`-polymorphism, effects, and subtyping into an annotated type and effect system for a fragment of Concurrent ML. First a small-step operational semantics is defined and next the annotated type and effect system is proved semantically sound. This provides insights into the rule for generalisation in the annotated type and effect system.

1 Introduction

In a recent paper [6] we developed an annotated type and effect system for a fragment of Concurrent ML. Judgements are of form

$$C, A \vdash e : \sigma \,\&\, b$$

with e an expression, σ a type or a type scheme, b a behaviour (recording channel allocations and thereby the set of "dangerous variables"), C a set of constraints among types and behaviours, and A an environment.

In this paper we address the soundness of the static semantics (i.e. the type system) wrt. a dynamic semantics. Statements of semantic soundness typically contain as premise that the inference system assigns a type t to e but the conclusion depends on the kind of dynamic semantics used: for a denotational semantics one may require (as in [4]) that the denotation of e "has type" t; for a big-step (natural) semantics one may require (as in [10, 3]) that if $e \rightarrow v$ then v "has type" t; for a small-step semantics [7] one requires (as in [11]) the following *subject reduction* property: if $e \rightarrow e'$ then the inference system also assigns e' the type t. In addition, in order to ensure that "well-typed programs do not go wrong" one must establish that "error configurations" (those which are "stuck") cannot be typed. This is in contrast to the development in [1] where the construction of a denotational semantics is *based on* an annotated type and effect system in that *only* well-typed programs are given a semantics.

We shall choose a small-step semantics as we consider this the most appropriate for concurrent languages; the configurations of the transition system will be process pools PP which map process identifiers into expressions. To get a flavour of how subject reduction is formulated in our setting consider the case where PP rewrites to PP' because process p allocates a fresh channel ch which is able to transmit values of type t', and suppose that

$C, A \vdash PP(p) : t \,\&\, b$

holds: then we must also have $C, A' \vdash PP'(p) : t \,\&\, b'$ where A' is as A except that ch is bound to t' `chan`, and where b is the "union" of the "current action" $\{t'\ \text{CHAN}\}$ (allocating a t'-channel) and the "future action" b'. The general picture is much as in [5] that types are unchanged whereas the behaviours get "smaller" and the environments are "extended".

Extending the environment is a potential danger to semantic soundness, cf. the considerations in [10, section 5] where it was concluded that store operations in Standard ML are harmless unless they actually expand the store. In [6] it was demonstrated that channel allocations (the way our setting "expands the store") may be harmful unless one is very careful when deciding the set of variables over which to generalise in the rule for `let` in the inference system: not only should this set be disjoint from the set of variables occurring in the behaviour (as is standard in effect systems, e.g. [9]) but it should also be "upwards closed" with respect to a constraint set. The present paper provides the formal justification and the proof of Lemma 37 highlights how the judicious choice of generalisation strategy allows to extend the environment.

Overview. We define a dynamic semantics which employs one system for the sequential components (Sect. 2.2) and another for the concurrent components (Sect. 2.3). Next (Sect. 2.4) we extend the repertoire of techniques [6] for normalising and manipulating the inference trees of the annotated type and effect system. Finally (Sect. 3), we show that the system is indeed semantically sound with respect to the dynamic semantics: we establish a sequential subject reduction result (Theorem 36) as a preparation for a concurrent subject reduction result (Theorem 41) which also demonstrates that the actions performed by the system are in a certain sense as "predicted" by the effect information; in Sect. 3.2 we demonstrate (informally) that it is not possible to assign a type to the "error configurations" which have been characterised in Proposition 22.

2 Inference System and Semantics

We shall make use of a variant of Concurrent ML [8] where expressions and constants are given by

$$\begin{aligned} e ::= {} & c \mid x \mid \texttt{fn}\ x \Rightarrow e \mid e_1\, e_2 \mid \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \\ & \mid \texttt{rec}\ f\ x \Rightarrow e \mid \texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \end{aligned}$$

$$\begin{aligned} c ::= {} & \texttt{()} \mid \texttt{true} \mid \texttt{false} \mid n \mid \texttt{pair} \mid \texttt{nil} \mid \texttt{cons} \\ & \mid \texttt{+} \mid \texttt{*} \mid \texttt{=} \mid \cdots \mid \texttt{fst} \mid \texttt{snd} \mid \texttt{hd} \mid \texttt{tl} \mid \texttt{isnil} \\ & \mid \texttt{send} \mid \texttt{receive} \\ & \mid \texttt{sync} \mid \texttt{channel} \mid \texttt{fork} \end{aligned}$$

where there are four kinds of constants: sequential constructors like `true` and `pair`, sequential base functions like `+` and `fst`, the non-sequential constructors `send` and `receive` for creating delayed output and input communications, and

the non-sequential base functions `sync` (for synchronising a delayed communication), `channel` (for allocating a new communication channel) and `fork` (for spawning a new process).

Types and behaviours are given by

$$\begin{aligned} t &::= \alpha \mid \texttt{unit} \mid \texttt{int} \mid \texttt{bool} \mid t_1 \times t_2 \mid t\ \texttt{list} \\ &\quad \mid\ t_1 \rightarrow^b\ t_2 \mid t\ \texttt{chan} \mid t\ \texttt{com}\ b \\ b &::= \{t\ \text{CHAN}\} \mid \beta \mid \emptyset \mid b_1 \cup b_2 \end{aligned}$$

where a type $t_1 \rightarrow^b\ t_2$ denotes a function which given a value of type t_1 computes a value of type t_2 with "side effect" b; where a type t `chan` denotes a channel allowing values of type t to be transmitted; and where a type t `com` b denotes a delayed communication that when synchronised will give rise to side effect b and result in a type t. A behaviour can (apart from the presence of behaviour variables) be viewed as a set of "atomic" behaviours which each record the creation of a channel.

Type schemes ts are of form $\forall(\vec{\alpha}\vec{\beta} : C).\ t$ with C a set of constraints, where a constraint is either of form $t_1 \subseteq t_2$ or of form $b_1 \subseteq b_2$. The type schemes of selected constants are given in Figure 1.

c	TypeOf(c)
`true`	`bool`
`pair`	$\forall(\alpha_1\alpha_2 : \emptyset).\ \alpha_1 \rightarrow^\emptyset\ \alpha_2 \rightarrow^\emptyset\ \alpha_1 \times \alpha_2$
`+`	$\texttt{int} \times \texttt{int} \rightarrow^\emptyset\ \texttt{int}$
`fst`	$\forall(\alpha_1\alpha_2 : \emptyset).\ \alpha_1 \times \alpha_2 \rightarrow^\emptyset\ \alpha_1$
`snd`	$\forall(\alpha_1\alpha_2 : \emptyset).\ \alpha_1 \times \alpha_2 \rightarrow^\emptyset\ \alpha_2$
`send`	$\forall(\alpha : \emptyset).\ (\alpha\ \texttt{chan}) \times \alpha \rightarrow^\emptyset\ (\alpha\ \texttt{com}\ \emptyset)$
`receive`	$\forall(\alpha : \emptyset).\ (\alpha\ \texttt{chan}) \rightarrow^\emptyset\ (\alpha\ \texttt{com}\ \emptyset)$
`sync`	$\forall(\alpha\beta : \emptyset).\ (\alpha\ \texttt{com}\ \beta) \rightarrow^\beta\ \alpha$
`channel`	$\forall(\alpha\beta : \{\{\alpha\ \text{CHAN}\} \subseteq \beta\}).\ \texttt{unit} \rightarrow^\beta\ (\alpha\ \texttt{chan})$
`fork`	$\forall(\alpha\beta : \emptyset).\ (\texttt{unit} \rightarrow^\beta\ \alpha) \rightarrow^\emptyset\ \texttt{unit}$

Fig. 1. Type schemes for selected constants.

Constructors actually construct something (that is, a composite type) and this construction takes place "silently":

Fact 1. Let c be a constructor (sequential or non-sequential), and let

$$t'_1 \rightarrow^{b'_1}\ \cdots t'_m \rightarrow^{b'_m}\ t' \text{ with } m \geq 0 \text{ and } t' \text{ not a function type}$$

be the unique "maximal decomposition" of (the type part of) TypeOf(c). Then

t' is not a type variable and $b'_i = \emptyset$ for all $i \in \{1 \ldots m\}$. □

The ordering among types and behaviours is defined in Figure 2; in particular notice that the ordering is contravariant in the argument position of a function type and that both t `chan` $\subseteq t'$ `chan` and $\{t \text{ CHAN}\} \subseteq \{t' \text{ CHAN}\}$ demand that $t \equiv t'$, i.e. $t \subseteq t'$ and $t' \subseteq t$, since t occurs covariantly when used in `receive` and contravariantly when used in `send`.

The inference system is defined in Figure 3. The rules for abstraction and application are as usual in effect systems: the latent behaviour of the body of a function abstraction is placed on the arrow of the function type, and once the function is applied the latent behaviour is added to the effect of evaluating the function and its argument, reflecting that the language is call-by-value. The rule for recursion makes use of function abstraction to concisely represent the "fixed point requirement" of typing recursive functions; note that we do not admit polymorphic recursion. The rule (sub) makes use of the subtyping and subeffecting relation defined in Fig. 2. The rule (ins) allows one to instantiate a type scheme and employs the notion of solvability: we say that the type scheme $\forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0$ is *solvable* from C by S_0 if $Dom(S_0) \subseteq \{\vec{\alpha}\vec{\beta}\}$ and if $C \vdash S_0\, C_0$, where we write $C \vdash C'$ to mean that[1] $C \vdash g_1 \subseteq g_2$ for all $g_1 \subseteq g_2$ in C'. The generalisation rule (gen) employs the notion of well-formedness, to be explained below.

A constraint set is *well-formed* if all constraints are of form $t \subseteq \alpha$ or $b \subseteq \beta$; this is motivated by the desire to be able to use the subtyping rules "backwards" (as spelled out in Lemma 4 below) and in ensuring that subeffecting preserves certain relations between the variables (see Lemma 5 below).

In order to define well-formedness for type schemes we need several auxiliary concepts: the judgement $C \vdash \gamma_1 \leftarrow \gamma_2$ holds if there exists $(g_1 \subseteq g_2)$ in C such that $\gamma_i \in FV(g_i)$ for $i = 1, 2$; we use $\leftarrow^*$ for the reflexive and transitive closure of $\leftarrow$; and we define *upwards closure* and *downwards closure* as follows:

$$X^{C\uparrow} = \{\gamma \mid \exists \gamma' \in X : C \vdash \gamma' \leftarrow^* \gamma\}$$
$$X^{C\downarrow} = \{\gamma \mid \exists \gamma' \in X : C \vdash \gamma \leftarrow^* \gamma'\}$$

Definition 2. A type scheme $\forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0$ is well-formed if C_0 is well-formed and if all constraints in C_0 contain at least one variable among $\{\vec{\alpha}\vec{\beta}\}$ and if it is *upwards closed*: that is $\{\vec{\alpha}\vec{\beta}\}^{C_0\uparrow} = \{\vec{\alpha}\vec{\beta}\}$.

Requiring a type scheme to be well-formed (in particular upwards closed) is a crucial feature in our approach to achieve a semantically sound system, cf. the discussion in the Introduction (and the proof of Lemma 37). The following trivial result proves useful:

Fact 3. Suppose $C \cup C_0 \vdash \gamma_1 \leftarrow \gamma_2$ with $\gamma_1 \notin FV(C)$. Then $C_0 \vdash \gamma_1 \leftarrow \gamma_2$.

[1] Following [6] we use g to stand for t or b and we use γ to stand for α or β and we use σ to stand for t or ts.

Ordering on behaviours

$$\text{(axiom)}\quad C \vdash b_1 \subseteq b_2 \qquad \text{if } (b_1 \subseteq b_2) \in C$$

$$\text{(refl)}\quad C \vdash b \subseteq b$$

$$\text{(trans)}\quad \frac{C \vdash b_1 \subseteq b_2 \qquad C \vdash b_2 \subseteq b_3}{C \vdash b_1 \subseteq b_3}$$

$$\text{(CHAN)}\quad \frac{C \vdash t \equiv t'}{C \vdash \{t \text{ CHAN}\} \subseteq \{t' \text{ CHAN}\}}$$

$$(\emptyset)\quad C \vdash \emptyset \subseteq b$$

$$(\cup)\quad C \vdash b_i \subseteq (b_1 \cup b_2) \qquad \text{for } i = 1, 2$$

$$\text{(lub)}\quad \frac{C \vdash b_1 \subseteq b \qquad C \vdash b_2 \subseteq b}{C \vdash (b_1 \cup b_2) \subseteq b}$$

Ordering on types

$$\text{(axiom)}\quad C \vdash t_1 \subseteq t_2 \qquad \text{if } (t_1 \subseteq t_2) \in C$$

$$\text{(refl)}\quad C \vdash t \subseteq t$$

$$\text{(trans)}\quad \frac{C \vdash t_1 \subseteq t_2 \qquad C \vdash t_2 \subseteq t_3}{C \vdash t_1 \subseteq t_3}$$

$$(\rightarrow)\quad \frac{C \vdash t_1' \subseteq t_1 \qquad C \vdash t_2 \subseteq t_2' \qquad C \vdash b \subseteq b'}{C \vdash (t_1 \rightarrow^{b} t_2) \subseteq (t_1' \rightarrow^{b'} t_2')}$$

$$(\times)\quad \frac{C \vdash t_1 \subseteq t_1' \qquad C \vdash t_2 \subseteq t_2'}{C \vdash (t_1 \times t_2) \subseteq (t_1' \times t_2')}$$

$$(\texttt{list})\quad \frac{C \vdash t \subseteq t'}{C \vdash (t\ \texttt{list}) \subseteq (t'\ \texttt{list})}$$

$$(\texttt{chan})\quad \frac{C \vdash t \equiv t'}{C \vdash (t\ \texttt{chan}) \subseteq (t'\ \texttt{chan})}$$

$$(\texttt{com})\quad \frac{C \vdash t \subseteq t' \qquad C \vdash b \subseteq b'}{C \vdash (t\ \texttt{com}\ b) \subseteq (t'\ \texttt{com}\ b')}$$

Fig. 2. Subtyping and subeffecting.

(con) $C, A \vdash c \,:\, \mathrm{TypeOf(c)} \,\&\, \emptyset$

(id) $C, A \vdash x \,:\, A(x) \,\&\, \emptyset$

(abs) $$\frac{C, A[x : t_1] \vdash e \,:\, t_2 \,\&\, b}{C, A \vdash \mathtt{fn}\ x \Rightarrow e \,:\, (t_1 \rightarrow^b\ t_2) \,\&\, \emptyset}$$

(app) $$\frac{C_1, A \vdash e_1 \,:\, (t_2 \rightarrow^b\ t_1) \,\&\, b_1 \qquad C_2, A \vdash e_2 \,:\, t_2 \,\&\, b_2}{(C_1 \cup C_2), A \vdash e_1\, e_2 \,:\, t_1 \,\&\, (b_1 \cup b_2 \cup b)}$$

(let) $$\frac{C_1, A \vdash e_1 \,:\, ts_1 \,\&\, b_1 \qquad C_2, A[x : ts_1] \vdash e_2 \,:\, t_2 \,\&\, b_2}{(C_1 \cup C_2), A \vdash \mathtt{let}\ x = e_1\ \mathtt{in}\ e_2 \,:\, t_2 \,\&\, (b_1 \cup b_2)}$$

(rec) $$\frac{C, A[f : t] \vdash \mathtt{fn}\ x \Rightarrow e \,:\, t \,\&\, b}{C, A \vdash \mathtt{rec}\ f\ x \Rightarrow e \,:\, t \,\&\, b}$$

(if) $$\frac{C_0, A \vdash e_0 \,:\, \mathtt{bool} \,\&\, b_0 \qquad C_1, A \vdash e_1 \,:\, t \,\&\, b_1 \qquad C_2, A \vdash e_2 \,:\, t \,\&\, b_2}{(C_0 \cup C_1 \cup C_2), A \vdash \mathtt{if}\ e_0\ \mathtt{then}\ e_1\ \mathtt{else}\ e_2 \,:\, t \,\&\, (b_0 \cup b_1 \cup b_2)}$$

(sub) $$\frac{C, A \vdash e \,:\, t \,\&\, b}{C, A \vdash e \,:\, t' \,\&\, b'} \qquad \text{if } C \vdash t \subseteq t' \text{ and } C \vdash b \subseteq b'$$

(ins) $$\frac{C, A \vdash e \,:\, \forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0 \,\&\, b}{C, A \vdash e \,:\, S_0\, t_0 \,\&\, b} \quad \text{if } \forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0 \text{ is solvable from } C \text{ by } S_0$$

(gen) $$\frac{C \cup C_0, A \vdash e \,:\, t_0 \,\&\, b}{C, A \vdash e \,:\, \forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0 \,\&\, b} \quad \text{if } \forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0 \text{ is both well-formed, solvable from } C, \text{ and satisfies } \{\vec{\alpha}\vec{\beta}\} \cap FV(C, A, b) = \emptyset$$

Fig. 3. The type inference system.

2.1 Properties of the Inference System

In this section we list some basic concepts and results which we shall use in the later development; except for Fact 8 and Fact 9 their proofs are given in [6].

The subtyping rules can be used "backwards" if the constraint set is well-formed:

Lemma 4. Suppose C is well-formed and that $C \vdash t \subseteq t'$.

- If $t' = t'_1 \rightarrow^{b'}\ t'_2$ there exist t_1, t_2 and b such that $t = t_1 \rightarrow^b\ t_2$ and such that $C \vdash t'_1 \subseteq t_1$, $C \vdash t_2 \subseteq t'_2$ and $C \vdash b \subseteq b'$.

– If $t' = t'_1$ `com` b' there exist t_1 and b such that $t = t_1$ `com` b and such that $C \vdash t_1 \subseteq t'_1$ and $C \vdash b \subseteq b'$.
– If $t' = t'_1 \times t'_2$ there exist t_1 and t_2 such that $t = t_1 \times t_2$ and such that $C \vdash t_1 \subseteq t'_1$ and $C \vdash t_2 \subseteq t'_2$.
– If $t' = t'_1$ `chan` there exists t_1 such that $t = t_1$ `chan` and such that $C \vdash t_1 \equiv t'_1$.
– If $t' = t'_1$ `list` there exists t_1 such that $t = t_1$ `list` and such that $C \vdash t_1 \subseteq t'_1$.
– If $t' =$ `int` (`bool`, `unit`) then $t =$ `int` (`bool`, `unit`).

Even if b is "less than" b' it does not necessarily hold that $FV(b) \subseteq FV(b')$, but taking downwards closure will establish the desired relation:

Lemma 5. Suppose that C is well-formed:

if $C \vdash b \subseteq b'$ then $FV(b)^{C\downarrow} \subseteq FV(b')^{C\downarrow}$.

We can apply a substitution to a judgement and still get a valid judgement (which even has the same shape, i.e. it is constructed using the same sequence of inference rules):

Lemma 6. For all substitutions S:

(a) If $C \vdash C'$ then $S\,C \vdash S\,C'$.
(b) If $C, A \vdash e : \sigma \,\&\, b$ then $S\,C, S\,A \vdash e : S\,\sigma \,\&\, S\,b$ (and has the same shape).

We can strengthen the constraint set and still get a valid judgement:

Lemma 7. For all sets C' of constraints satisfying $C' \vdash C$:

(a) If $C \vdash C_0$ then $C' \vdash C_0$.
(b) If $C, A \vdash e : \sigma \,\&\, b$ then $C', A \vdash e : \sigma \,\&\, b$ (and has the same shape).

We can swap distinct identifiers in the environment and still get a valid judgement:

Fact 8. Let x and y be distinct identifiers: if $C, A_1[x : \sigma_1][y : \sigma_2]A_2 \vdash e : \sigma \,\&\, b$ then $C, A_1[y : \sigma_2][x : \sigma_1]A_2 \vdash e : \sigma \,\&\, b$ (and has the same shape).

We can augment the environment with spurious identifiers and still get a valid judgement:

Fact 9. Let x be an identifier not occurring in e and let t be an arbitrary type. If $C, A \vdash e : \sigma \,\&\, b$ then $C, A[x : t] \vdash e : \sigma \,\&\, b$ (and has the same shape).

Proof. Let α be a fresh type variable. Then a straightforward induction in the proof tree (using Fact 8) tells us that $C, A[x : \alpha] \vdash e : \sigma \,\&\, b$ (and has the same shape). Now apply Lemma 6 with the substitution $[\alpha \mapsto t]$.

The proof of semantic soundness is complicated by the presence of the non-syntax directed rules (sub), (gen) and (ins) of Figure 3; this motivates considering inference trees having a more manageable shape.

Definition 10. An inference tree for $C, A \vdash e : t \,\&\, b$ is *T-normalised* if it is created by:

- (con) or (id); or
- (ins) applied to (con) or (id); or
- (abs), (app), (rec), (if) or (sub) applied to T-normalised inference trees; or
- (let) applied to a TS-normalised inference tree and a T-normalised inference.

An inference tree for $C, A \vdash e : ts \,\&\, b$ is *TS-normalised* if it is created by:

- (gen) applied to a T-normalised inference tree.

We shall write $C, A \vdash_n e : \sigma \,\&\, b$ if the inference tree is T-normalised (if σ is a type) or TS-normalised (if σ is a type scheme).

Notice that if $jdg = C, A \vdash e : \sigma \,\&\, b$ occurs in a normalised inference tree then jdg itself will be normalised, unless jdg is created by (con) or (id) and σ is a type scheme. By requiring an inference to be normalised one restricts the use of (ins), but the following result indicates that this is not a severe restriction.

Lemma 11. Suppose that

$$jdg = C, A \vdash e : S\,t_0 \,\&\, b$$

follows by an application of (ins) to the normalised judgement

$$jdg' = C, A \vdash_n e : \forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0 \,\&\, b$$

where $Dom(S) \subseteq \{\vec{\alpha}\vec{\beta}\}$ and $C \vdash S\,C_0$. Then also jdg has a normalised inference:

$$C, A \vdash_n e : S\,t_0 \,\&\, b.$$

Definition 12. An inference is contraint-saturated (indicated by subscript c) whenever for all occurrences of the rules (app), (let), and (if) it holds that all constraint sets in the premises equal the constraint set in the conclusion.

An inference is strongly normalised (indicated by subscript s) if it is normalised as well as constraint-saturated.

Fact 13. Given an inference tree for $C, A \vdash e : \sigma \,\&\, b$ there exists a constraint-saturated inference tree $C, A \vdash_c e : \sigma \,\&\, b$ (that has the same shape).

In particular, a normalised inference tree $C, A \vdash_n e : \sigma \,\&\, b$ can be transformed into one that is strongly normalised.

2.2 The Sequential Semantics

We are now going to define a small-step semantics for the sequential part of the language. Transitions take the form $e \rightarrow e'$ where e and e' are expressions that are *essentially closed*: this means that they may contain free channel identifiers ch (created by previous channel allocations) but that they must not contain any free program identifiers.

We first stipulate the semantics of the sequential base functions (`+`, `fst` etc.) by means of an "evaluation function" δ:

Definition 14. The function δ is a partial mapping from expressions into expressions (preserving the property of being essentially closed); the domain of δ is a subset of the expressions of form $c\,e$ with c a sequential base function. It is defined by the (incomplete) Figure 4; notice that we encode "runtime errors" such as $\texttt{hd}\,(\texttt{nil})$ as loops whereas e.g. $\texttt{hd}\,(7)$ is undefined.

c	e	$\delta(c\,e)$
`fst`	$\texttt{pair}\,e_1\,e_2$	e_1
`snd`	$\texttt{pair}\,e_1\,e_2$	e_2
`hd`	$\texttt{cons}\,e_1\,e_2$	e_1
`hd`	`nil`	`hd nil`
`tl`	$\texttt{cons}\,e_1\,e_2$	e_2
`tl`	`nil`	`tl nil`
`isnil`	`nil`	`true`
`isnil`	$\texttt{cons}\,e_1\,e_2$	`false`
`+`	$\texttt{pair}\,n_1\,n_2$	n where $n = n_1 + n_2$
⋮	⋮	
`/`	$\texttt{pair}\,n\,0$	$/\,(\texttt{pair}\,n\,0)$

Fig. 4. The evaluation function δ.

We next introduce the notion of *weakly evaluated expressions* ($w \in \mathit{WExp}$) that are the "terminal configurations" of the sequential semantics:

Definition 15. An expression w is a *weakly evaluated expression* provided that either

- w is a constant c; or
- w is a channel identifier ch; or
- w is a function abstraction $\texttt{fn}\ x \Rightarrow e$; or
- w is of form $c\,w_1 \cdots w_n$, where $n \geq 1$, where $w_1, \cdots, w_n$ are weakly evaluated expressions, and where c is a constructor (sequential or non-sequential).

To formalise the call-by-value evaluation strategy we shall as in [8, 5] employ the notion of *evaluation contexts*[2].

Definition 16. Evaluation contexts E take the form

$$E ::= [\,] \mid E\,e \mid w\,E \mid \texttt{let}\ x = E\ \texttt{in}\ e \mid \texttt{if}\ E\ \texttt{then}\ e_1\ \texttt{else}\ e_2$$

Notice that E is a context with exactly one hole in it, and that this hole is not inside the scope of any defining occurrence of a program identifier. We write $E[e]$ for the expression that has the hole in E replaced by e, and similarly $E[E']$ for the evalution context that results by replacing the hole in E with E'.

Fact 17. $(E_1[E_2])[e] = E_1[E_2[e]]$.

Proof. The proof is by induction in E_1. If $E_1 = [\,]$ the equation reads $E_2[e] = E_2[e]$, so assume that E_1 is a composite context and let us consider the case $E_1 = E\,e_2$ (the other cases are similar). By using the induction hypothesis for E we get the desired equation

$$E_1[E_2][e] = (E\,e_2)[E_2][e] = (E[E_2]\,e_2)[e] = E[E_2][e]\,e_2 = E[E_2[e]]\,e_2 = E_1[E_2[e]].$$

Now we are ready for:

Definition 18. *Sequential Evaluation*

The sequential transition relation $\rightarrow$ is defined by

$E[e] \rightarrow E[e']$ provided $e \rightharpoonup e'$ holds according to the following definition:

$$\begin{array}{rl}
\text{(apply) } (\texttt{fn}\ x \Rightarrow e)\,w & \rightharpoonup e[w/x] \\
\text{(delta) } c\,w & \rightharpoonup e' \text{ if } e' = \delta(c\,w) \\
\text{(let) } \texttt{let}\ x = w\ \texttt{in}\ e & \rightharpoonup e[w/x] \\
\text{(rec) } \texttt{rec}\ f\ x \Rightarrow e & \rightharpoonup (\texttt{fn}\ x \Rightarrow e)[(\texttt{rec}\ f\ x \Rightarrow e)/f] \\
\text{(branch) } \texttt{if}\ w\ \texttt{then}\ e_1\ \texttt{else}\ e_2 & \rightharpoonup \begin{cases} e_1 & \text{if } w = \texttt{true} \\ e_2 & \text{if } w = \texttt{false} \end{cases}
\end{array}$$

Fact 19. If $e \rightarrow e'$ with e essentially closed then also e' is essentially closed.

Observe that $e_1\,e_2 \rightarrow e'$ holds iff either (i) $e_1\,e_2 \rightharpoonup e'$, or (ii) there exists e_1' such that $e_1 \rightarrow e_1'$ and $e' = e_1'\,e_2$, or (iii) there exists e_2' such that $e_2 \rightarrow e_2'$ and $e' = e_1\,e_2'$ (in which case e_1 is a weakly evaluated expression). Further observe that $\texttt{let}\ x = e_1\ \texttt{in}\ e_2 \rightarrow e'$ holds iff either (i) $\texttt{let}\ x = e_1\ \texttt{in}\ e_2 \rightharpoonup e'$, or (ii) there exists e_1' such that $e_1 \rightarrow e_1'$ and $e' = \texttt{let}\ x = e_1'\ \texttt{in}\ e_2$. Finally observe that $\texttt{if}\ e_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \rightarrow e'$ holds iff either (i) $\texttt{if}\ e_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \rightharpoonup e'$, or (ii) there exists e_0' such that $e_0 \rightarrow e_0'$ and $e' = \texttt{if}\ e_0'\ \texttt{then}\ e_1\ \texttt{else}\ e_2$.

As expected we have:

Fact 20. If w is a weakly evaluated expression then $w \not\rightarrow$.

Proof. It is easy to see that $w \not\rightharpoonup$; the result then follows by induction on w. □

We shall say that an essentially closed expression e is *exhausted* if it is not weakly evaluated and yet $e \not\rightarrow$. We shall say that an exhausted expression e is *top-level exhausted* if it cannot be written on the form $e = E[e']$ with $E \neq [\,]$ and with e' exhausted. It is easy to see (using Fact 17) that for any exhausted expression e there exists E and top-level exhausted e' such that $e = E[e']$.

Fact 21. Suppose that e is top-level exhausted; then either

- $e = c\,w$ with c a non-sequential base function; or
- $e = c\,w$ with c a sequential base function where $\delta(e)$ is undefined; or
- $e = ch\,w$ with ch a channel identifier; or

– $e = \texttt{if}\ w\ \texttt{then}\ e_1\ \texttt{else}\ e_2$ with $w \notin \{\texttt{true}, \texttt{false}\}$.

Proof. We perform a case analysis on e (which is essentially closed). If e is a constant, a channel identifier or an abstraction then e is weakly evaluated and hence not exhausted. If e is of form $\texttt{rec}\ f\ x \Rightarrow e$, then $e \rightharpoonup \cdots$ and hence e is not exhausted.

If e is of form $\texttt{let}\ x = e_1\ \texttt{in}\ e_2$ then e_1 is essentially closed and $e_1 \not\rightarrow$ (as otherwise $e \rightarrow$) but e_1 is not exhausted (as e is top-level exhausted). Hence we conclude that e_1 is weakly evaluated, but this is a contradiction since then $e \rightharpoonup \cdots$.

If e is of form $\texttt{if}\ e_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2$ then e_0 is essentially closed and $e_0 \not\rightarrow$ (as otherwise $e \rightarrow$) but e_0 is not exhausted (as e is top-level exhausted). Hence we conclude that e_0 is weakly evaluated; and this yields the claim since if $e_0 = \texttt{true}$ or $e_0 = \texttt{false}$ then $e \rightharpoonup \cdots$.

If e is of form $e_1\, e_2$ we infer (using the same technique as in the above two cases) that e_1 is a weakly evaluated expression w_1 and subsequently that e_2 is a weakly evaluated expression w_2. Since e is not a weakly evaluated expression it cannot be the case that w_1 is of form $c\, w'_1 \cdots w'_n$ with c a constructor and with $n \geq 0$; and since $e \not\rightarrow$ it cannot be the case that w_1 is of form $\texttt{fn}\ x \Rightarrow e'_1$ or a sequential base function such that $\delta(e)$ is defined. This yields the claim. □

From the preceding results we get:

Proposition 22. Suppose that e is essentially closed and that $e \rightarrow^* e' \not\rightarrow$. Then either

1. e' is a weakly evaluated expression; or
2. e' is of form $E[c\, w]$ with c a non-sequential base function; or
3. e' is either of form $E[c\, w]$ with c a sequential base function where $\delta(c\, w)$ is undefined (e.g. $\texttt{hd}\ 7$), or of form $E[ch\, w]$, or of form $E[\texttt{if}\ w\ \texttt{then}\ e_1\ \texttt{else}\ e_2]$ with $w \notin \{\texttt{true}, \texttt{false}\}$.

The configurations listed in case 3 can be thought of as error configurations, whereas in Section 2.3 we shall see that case 2 corresponds to a process that may be able to perform a concurrent action.

Fact 23. The rewriting relation $\rightarrow$ is deterministic.

Proof. We perform induction on e to show that if $e \rightarrow e'$ and $e \rightarrow e''$ then $e' = e''$. If e is a constant, a variable or a function abstraction then $e \not\rightarrow$ and if e is of form $\texttt{rec}\ f\ x \Rightarrow e$ determinism is obvious.

If e is of form $\texttt{let}\ x = w\ \texttt{in}\ e_2$ the claim follows from $w \not\rightarrow$. If e is of form $\texttt{let}\ x = e_1\ \texttt{in}\ e_2$ with e_1 not a weakly evaluated expression then e' takes the form $\texttt{let}\ x = e'_1\ \texttt{in}\ e_2$ where $e_1 \rightarrow e'_1$ and by the induction hypothesis this e'_1 is unique.

If e is of form $\texttt{if}\ w\ \texttt{then}\ e_1\ \texttt{else}\ e_2$ the claim follows from $w \not\rightarrow$. If e is of form $\texttt{if}\ e_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2$ with e_0 not a weakly evaluated expression then e' takes the form $\texttt{if}\ e'_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2$ where $e_0 \rightarrow e'_0$ and by the induction hypothesis this e'_0 is unique.

We are left with the case $e = e_1\, e_2$. First suppose that e_1 is not weakly evaluated. Then $e \not\rightarrow$ and we infer that e' takes the form $e_1'\, e_2$ where $e_1 \rightarrow e_1'$ so by the induction hypothesis this e_1' is unique.

Next suppose that $e = w_1\, e_2$ with e_2 not weakly evaluated. Then $e \not\rightarrow$ and as $w_1 \not\rightarrow$ we infer that e' takes the form $w_1\, e_2'$ where $e_2 \rightarrow e_2'$ so by the induction hypothesis this e_2' is unique.

Finally assume that $e = w_1\, w_2$. Then $w_1 \not\rightarrow$ and $w_2 \not\rightarrow$ so it must hold that $e \rightarrow e'$. If w_1 is a function abstraction this e' is clearly unique; and if w_1 is a sequential base function uniqueness follows from δ being a function. □

2.3 The Concurrent Semantics

Next we are going to define a small-step semantics for the concurrent part of the language. Transitions take the form $PP \xrightarrow{a} PP'$, where PP as well as PP' is a *process pool* which is a finite mapping from process identifiers p into essentially closed expressions, and where a is a label describing what kind of action is taken.

Definition 24. *Concurrent Evaluation*

The concurrent transition relation $\xrightarrow{a}$ is defined by:

$$PP[p : e] \xrightarrow{\texttt{seq}} PP[p : e'] \quad \text{if } e \rightarrow e'$$

$$PP[p : E[\texttt{channel ()}]] \xrightarrow{p\ \texttt{chan}\ ch} PP[p : E[ch]] \quad \text{if } ch \text{ not in } PP \text{ or } E$$

$$PP[p : E[\texttt{fork}\, w]] \xrightarrow{p\ \texttt{fork}\ p'} PP[p : E[()]][p' : w\ ()] \quad \text{if } p' \notin Dom(PP) \cup \{p\}$$

$$PP[p_1 : E_1[\texttt{sync}\,(\texttt{send}\,(\texttt{pair}\, ch\, w))]]\ [p_2 : E_2[\texttt{sync}\,(\texttt{receive}\, ch)]] \xrightarrow{p_1\ \texttt{comm}\ p_2} PP[p_1 : E_1[w]][p_2 : E_2[w]] \quad \text{if } p_1 \neq p_2$$

2.4 Reasoning about Proof Trees

In this section we present some auxiliary results which will eventually enable us to show that if there is a typing for e and if e gets "rewritten" into e' (sequentially or concurrently) then we can construct a typing for e'.

A common pattern will be that we have some judgement $C', A' \vdash E[e] : \sigma' \,\&\, b'$, but we want to reason about the typing of e rather than that of $E[e]$. To this end we need to be precise about what it means for a judgement to occur "at the address indicated by the hole in E":

Definition 25. The judgement $jdg = (C, A \vdash e : \sigma \,\&\, b)$ occurs at E (with depth n) in the inference tree for the judgement $jdg' = (C', A' \vdash e' : \sigma' \,\&\, b')$, provided that *either*

- $jdg = jdg'$ and $E = [\,]$ (and $n = 0$); *or*
- there exists a judgement jdg'' and an evaluation context E'' such that jdg occurs at E'' (with depth $n-1$) in the inference tree for jdg'', and such that the last rule applied in the inference tree for jdg' is *either*
  - (sub), (ins), or (gen), with jdg'' as premise and with $E = E''$; *or*
  - (app), with jdg'' as leftmost premise and with $E = E''\, e_2$ where e' is of form $e_1\, e_2$; *or*
  - (app), with jdg'' as rightmost premise and with $E = w_1\, E''$ where e' is of form $w_1\, e_2$; *or*
  - (let), with jdg'' as leftmost premise and with $E = \texttt{let}\ x = E''\ \texttt{in}\ e_2$ where e' is of form $\texttt{let}\ x = e_1\ \texttt{in}\ e_2$; *or*
  - (if), with jdg'' as leftmost premise and with $E = \texttt{if}\ E''\ \texttt{then}\ e_1\ \texttt{else}\ e_2$ where e' is of form $\texttt{if}\ e_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2$.

This is well-defined in the size of the inference tree for jdg'. As expected we have the following results, the latter to be proved in Appendix A:

Fact 26. Suppose that $C, A \vdash e : \sigma \,\&\, b$ occurs at E in the inference tree for $C', A' \vdash e' : \sigma' \,\&\, b'$; then $e' = E[e]$.

Fact 27. Given $jdg' = (C', A' \vdash E[e] : \sigma' \,\&\, b')$; then there exists (at least one) judgement jdg of form $C, A \vdash e : \sigma \,\&\, b$ such that jdg occurs at E in the inference tree for jdg'. If jdg' is (strongly) normalised we can assume that jdg is (strongly) normalised.

The following result is convenient when performing induction in the depth of a judgement in an inference tree (as we shall do in the proof of Lemma 37):

Fact 28. Suppose the judgement jdg occurs at E with depth n in the inference tree for jdg', where $n \geq 2$. Then there exists a judgement jdg'' and evaluation contexts E_1 and E_2 such that

> jdg occurs at E_1 with depth $< n$ in the inference tree for jdg''; and
> jdg'' occurs at E_2 with depth $< n$ in the inference tree for jdg'; and
> $E = E_2[E_1]$.

Moreover, if jdg' is (strongly) normalised we can assume that also jdg'' is (strongly) normalised.

Proof. We can clearly use jdg'' as in Definition 25. □

Having set up the necessary machinery we are now ready for the first result, which states that "equivalent" expressions may be substituted for each other:

Fact 29. Suppose the judgement $C, A \vdash e : \sigma \,\&\, b$ occurs at E in the inference tree of $C', A' \vdash E[e] : \sigma' \,\&\, b'$. If e_n is such that $C, A \vdash e_n : \sigma \,\&\, b$ then also

$C', A' \vdash E[e_n] \,:\, \sigma' \,\&\, b'$

and this judgement is normalised if the abovementioned judgements are.

It proves useful to know something about the relationship between the root of an inference tree and the interior nodes of the tree; since the hole in an evaluation context is not inside the scope of any bound identifier we have:

Fact 30. Suppose the judgement $C, A \vdash e \,:\, \sigma \,\&\, b$ occurs at E in the inference tree of $C', A' \vdash e' \,:\, \sigma' \,\&\, b'$. Then

$A' = A$ and if C' is well-formed then also C is well-formed. □

The following lemma tells us something about the relationship between the type of an expression $c\, e_1 \cdots e_n$, the type of c, and the type of each e_i:

Lemma 31. Suppose that C is well-formed and that

$C, A \vdash_n \; c\, e_1 \cdots e_n \,:\, t \,\&\, b \;\; (n \geq 0)$

and that TypeOf(c) is of form

$\forall(\vec{\alpha}\vec{\beta} : C_0).\; t'_1 \to^{b'_1} \; \cdots t'_m \to^{b'_m} \; t''$

and that if c is a base function then $m \geq n$.

Then in all cases (i.e. also if c is a constructor) we can write

$\text{TypeOf}(c) = \forall(\vec{\alpha}\vec{\beta} : C_0).\; t'_1 \to^{b'_1} \; \cdots t'_n \to^{b'_n} \; t'$

and there exists S, $t_1 \cdots t_n$, and $b_1 \cdots b_n$, such that

$Dom(S) \subseteq \{\vec{\alpha}\vec{\beta}\}$ and $C \vdash S\, C_0$ and $C \vdash S\, t' \subseteq t$;

for all $i \in \{1 \cdots n\}$: $C, A \vdash_s \; e_i \,:\, t_i \,\&\, b_i$ and $C \vdash t_i \subseteq S\, t'_i$ and $C \vdash b_i \subseteq b$ and $C \vdash S\, b'_i \subseteq b$.

Similarly, if $\text{TypeOf}(c) = t'_1 \to^{b'_1} \; \cdots t'_m \to^{b'_m} \; t''$ in which case $\{\vec{\alpha}\vec{\beta}\} = \emptyset$ and $C_0 = \emptyset$ (so we have $S = \text{Id}$).

Proof. See Appendix A.

The following two lemmas, both to be proved in Appendix A, show

- that we can replace variables by expressions of the same type, provided these expressions have an empty behaviour; and
- that the latter condition can always be obtained for weakly evaluated expressions.

Lemma 32. Suppose that $C, A[x : \sigma'] \vdash_n \; e \,:\, \sigma \,\&\, b$ and $C, A \vdash_n \; e' \,:\, \sigma' \,\&\, \emptyset$; then $C, A \vdash_n \; e[e'/x] \,:\, \sigma \,\&\, b$.

Lemma 33. Suppose that $C, A \vdash_n \; w \,:\, \sigma \,\&\, b$ with C well-formed; then

$C, A \vdash_n \; w \,:\, \sigma \,\&\, \emptyset$.

3 Semantic Soundness

In this section we shall prove that the sequential as well as the concurrent transition relation "preserves types" and "decreases behaviours". First an auxiliary concept:

Definition 34. An environment A is a *channel environment* if $Dom(A)$ is a subset of the channel identifiers and for each $ch \in Dom(A)$ that $A(ch)$ takes the form t `chan`.

We then *impose* that the concurrent transition relation only operates on channel environments. This is going to hold for the initial environment which is empty, and we shall see that the concurrent soundness result (Theorem 41) guarantees that the assumption is maintained; thus our decision seems to be a benign one. To see that it is actually necessary to impose the condition, note that otherwise the type of the channel would be polymorphic and the sender and receiver of a transmitted value would then be allowed to disagree on its type; this is exactly where type insecurities would creep in.

3.1 Sequential Soundness

First we shall prove that "top-level" reduction is sound:

Lemma 35. Let C be well-formed and let A be a channel environment. If $e \longrightarrow e'$ and

$$C, A \vdash_n e : \sigma \,\&\, b$$

then also

$$C, A \vdash_n e' : \sigma \,\&\, b.$$

Proof. By Fact 13 we can assume that we even have $C, A \vdash_s e : \sigma \,\&\, b$; we perform induction in the proof tree.

The rule (gen) has been applied: Then the situation is

$$\frac{C \cup C_0, A \vdash_s e : t_0 \,\&\, b}{C, A \vdash_s e : \forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0 \,\&\, b}\ \text{(gen)}$$

and as C_0 is well-formed we can apply the induction hypothesis to get

$$C \cup C_0, A \vdash_n e' : t_0 \,\&\, b$$

from which we by (gen) arrive at the desired judgement

$$C, A \vdash_n e' : \forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0 \,\&\, b.$$

The rule (sub) has been applied: Then the situation is

$$\frac{C, A \vdash_s e : t \,\&\, b}{C, A \vdash_s e : t' \,\&\, b'} \text{ (sub)}$$

and the induction hypothesis yields

$$C, A \vdash_n e' : t \,\&\, b$$

from which we by (sub) arrive at the desired judgement

$$C, A \vdash_n e' : t' \,\&\, b'.$$

Otherwise a "structural" rule has been applied (for if (ins) has been applied then e is a constant or an identifier and then $e \not\rightharpoonup$); we now perform case analysis on the transition $\rightharpoonup$:

The transition (let) has been applied: Then the situation is

$$\frac{C, A \vdash_s w : ts \,\&\, b_1 \qquad C, A[x : ts] \vdash_s e : t \,\&\, b_2}{C, A \vdash \texttt{let}\ x = w\ \texttt{in}\ e : t \,\&\, b_1 \cup b_2} \text{ (let)}$$

and using Lemma 33 we have

$$C, A \vdash_n w : ts \,\&\, \emptyset$$

which by Lemma 32 can be combined with the second premise of the inference to yield

$$C, A \vdash_n e[w/x] : t \,\&\, b_2$$

and since $C \vdash b_2 \subseteq (b_1 \cup b_2)$ we can apply (sub) to get the desired result.

The transition (rec) has been applied: Then the situation is

$$\frac{C, A[f : t] \vdash_s \texttt{fn}\ x \Rightarrow e : t \,\&\, b}{C, A \vdash \texttt{rec}\ f\ x \Rightarrow e : t \,\&\, b} \text{ (rec)}$$

and using Lemma 33 we have

$$C, A[f : t] \vdash_n \texttt{fn}\ x \Rightarrow e : t \,\&\, \emptyset$$

so by applying (rec) we get the judgement

$$C, A \vdash_n \texttt{rec}\ f\ x \Rightarrow e : t \,\&\, \emptyset$$

which by Lemma 32 can be combined with the premise of the inference to yield

$$C, A \vdash_n (\texttt{fn}\ x \Rightarrow e)[(\texttt{rec}\ f\ x \Rightarrow e)/f] : t \,\&\, b$$

which is as desired.

The transition (branch) has been applied: Then the situation is

$$\frac{C, A \vdash_s w : \texttt{bool} \,\&\, b_0 \qquad C, A \vdash_s e_1 : t \,\&\, b_1 \qquad C, A \vdash_s e_2 : t \,\&\, b_2}{C, A \vdash \texttt{if}\ w\ \texttt{then}\ e_1\ \texttt{else}\ e_2 : t \,\&\, b_0 \cup (b_1 \cup b_2)}\ (\text{if})$$

The claim now follows from the fact that for $i = 1, 2$ we have $C \vdash b_i \subseteq b_0 \cup (b_1 \cup b_2)$.

The transition (apply) has been applied: Then the situation is

$$\frac{\dfrac{C, A[x : t_2'] \vdash_s e : t' \,\&\, b_0'}{C, A \vdash_s \texttt{fn}\ x \Rightarrow e : t_2 \to^{b_0} t \,\&\, b_1}\ (\text{abs})\ (\text{sub})^* \qquad C, A \vdash_s w : t_2 \,\&\, b_2}{C, A \vdash (\texttt{fn}\ x \Rightarrow e)\, w : t \,\&\, (b_1 \cup b_2 \cup b_0)}\ (\text{app})$$

where $C \vdash t_2' \to^{b_0'} t' \subseteq t_2 \to^{b_0} t$ which by Lemma 4 implies that

$C \vdash t_2 \subseteq t_2'$ and $C \vdash b_0' \subseteq b_0$ and $C \vdash t' \subseteq t$.

By Lemma 33 followed by an application of (sub) we get

$C, A \vdash_n w : t_2' \,\&\, \emptyset$

which by Lemma 32 can be combined with the upmost leftmost premise of the inference to yield

$C, A \vdash_n e[w/x] : t' \,\&\, b_0'$

and since $C \vdash t' \subseteq t$ and $C \vdash b_0' \subseteq b_0 \subseteq b_1 \cup b_2 \cup b_0$ we can apply (sub) to get the desired result.

The transition (delta) has been applied: The claim then follows from an examination of Figure 4; below we shall list a typical case only. In all cases we make use of Lemma 31 and Lemma 4 which can be applied since C is well-formed.

$e = \texttt{fst}\,(\texttt{pair}\, w_1\, w_2)$ and $\delta(e) = w_1$: Then the situation is that

$C, A \vdash_s \texttt{fst}\,(\texttt{pair}\, w_1\, w_2) : t \,\&\, b$

so since TypeOf(`fst`) $= \forall(\alpha_1\alpha_2 : \emptyset).\ \alpha_1 \times \alpha_2 \to^{\emptyset} \alpha_1$, Lemma 31 tells us that there exists t_0, b_0 and S_0 such that

$C, A \vdash_s \texttt{pair}\, w_1\, w_2 : t_0 \,\&\, b_0$ and

$C \vdash S_0\, \alpha_1 \subseteq t$ and $C \vdash t_0 \subseteq S_0\, (\alpha_1 \times \alpha_2)$ and $C \vdash b_0 \subseteq b$.

Since TypeOf(`pair`) $= \forall(\alpha_1\alpha_2 : \emptyset).\ \alpha_1 \to^{\emptyset} \alpha_2 \to^{\emptyset} \alpha_1 \times \alpha_2$, Lemma 31 tells us that there exists t_1, b_1, t_2, b_2 and S such that

$C, A \vdash_s w_1 : t_1 \,\&\, b_1$ and $C, A \vdash_s w_2 : t_2 \,\&\, b_2$;
$C \vdash t_1 \subseteq S\,\alpha_1$ and $C \vdash t_2 \subseteq S\,\alpha_2$ and $C \vdash b_1 \cup b_2 \subseteq b_0$;
$C \vdash S\,(\alpha_1 \times \alpha_2) \subseteq t_0$.

We thus have

$$C \vdash t_1 \times t_2 \subseteq S\,\alpha_1 \times S\,\alpha_2 \subseteq t_0 \subseteq S_0\,\alpha_1 \times S_0\,\alpha_2$$

so from Lemma 4 we deduce that

$$C \vdash t_1 \subseteq S_0\,\alpha_1 \subseteq t$$

and since $C \vdash b_1 \subseteq b_1 \cup b_2 \subseteq b_0 \subseteq b$ we from $C, A \vdash_s w_1 : t_1 \,\&\, b_1$ get

$$C, A \vdash_n w_1 : t \,\&\, b.$$

which is as desired. □

Theorem 36. *Sequential soundness*
Let C be well-formed and let A be a channel environment. If $e_1 \rightarrow e_2$ and

$$C, A \vdash_n e_1 : \sigma \,\&\, b$$

then also

$$C, A \vdash_n e_2 : \sigma \,\&\, b.$$

Proof. There exists E, e_1' and e_2' such that

$$e_1 = E[e_1'] \text{ and } e_2 = E[e_2'] \text{ and } e_1' \rightarrowtail e_2'.$$

By Fact 27 there exists C', A', σ' and b' such that $C', A' \vdash_n e_1' : \sigma' \,\&\, b'$ occurs at E in the inference tree of $C, A \vdash_n E[e_1'] : \sigma \,\&\, b$. By Fact 30 we infer that $A' = A$ and that C' is well-formed; this enables us to use Lemma 35 from which we get

$$C', A' \vdash_n e_2' : \sigma' \,\&\, b'$$

and by Fact 29 we get the desired judgement

$$C, A \vdash_n E[e_2'] : \sigma \,\&\, b.$$

This completes the proof.

3.2 Erroneous Programs cannot be Typed

The purpose of types is to detect certain kinds of errors at analysis time rather than at execution time. To this end one usually (cf. the methodical considerations in [11]) wants a result that guarantees that "error configurations are not typeable"; here we presuppose some well-formed constraint set C and some channel environment A. By Proposition 22 and the discussion after it (together with Fact 27 and Fact 30), it suffices to consider each of the error-configurations listed below, and to show that it is not typeable; for this we make use of Lemma 4.

$ch\,w$ with ch a channel identifier: since $A(ch)$ is of form t `chan` in order for $ch\,w$ to be typeable it must be the case that $C \vdash t$ `chan` $\subseteq t_1 \to^b t_2$ for some t_1,t_2; this conflicts with Lemma 4.

`if` w `then` e_1 `else` e_2 with $w \notin \{$`true`, `false`$\}$: for this to be typeable it must hold that

> w can be assigned the type `bool`.

From Lemma 4 we infer that w cannot be a channel identifier (as A is a channel environment); w cannot be a function abstraction; and an examination of Figure 1 (using Lemma 31) will reveal that w cannot be a constant apart from `true`, `false` or of form $c\,w_1 \cdots w_n$ ($n \geq 1$) with c a constructor.

$c\,w$ with c a sequential base function with $\delta(c\,w)$ undefined: consider e.g. the expression `fst` w. For this to be typeable there must (cf. Lemma 31) exist t_1 and t_2 such that

> w can be assigned the type $t_1 \times t_2$.

From Lemma 4 we infer that w cannot be a channel identifier (as A is a channel environment); w cannot be a function abstraction; and an examination of Figure 1 (using Lemma 31) will reveal that w cannot be a constant and that w cannot be of form $c\,w_1 \cdots w_n$ with c a constructor apart from `pair`. Thus w is of form `pair` $w_1\,w_2$, but then $\delta(c\,w)$ is not undefined.

3.3 Concurrent Soundness

First a crucial result which generalises Fact 29 by allowing the "new" expression e_n to be typed using a channel environment which is an *extension* of the channel environment in which the old expression e was typed. Such an extension is a potential danger to semantic soundness, cf. the remarks made in the Introduction; in order to construct an inference tree with the new environment we must demand that the new environment variables are "present" in the behaviour.

Lemma 37. Suppose the judgement $jdg = (C, A \vdash e : \sigma \,\&\, b)$ occurs at E in the strongly normalised inference $jdg' = (C', A \vdash_s E[e] : \sigma' \,\&\, b')$ where C' (and by Fact 30 then also C) is well-formed.

Let b_n be a behaviour and let A_n be of form $A[x_1 : \sigma_1][\cdots][x_m : \sigma_m]$ with $m \geq 0$, such that $x_1 \cdots x_m$ do not occur in $E[e]$ and such that $FV(\sigma_1) \cup \cdots \cup FV(\sigma_m) \subseteq FV(b_n)$.

Let e_n be an expression and b_r a behaviour such that

$$C, A_n \vdash_n e_n : \sigma \,\&\, b_r \text{ and}$$
$$C \vdash b_n \cup b_r \subseteq b.$$

Then there exists b'_r such that

$$C', A_n \vdash_n E[e_n] : \sigma' \,\&\, b'_r \text{ and}$$
$$C' \vdash b_n \cup b'_r \subseteq b'.$$

Moreover, there exists S with $Dom(S) \cap FV(A, b_n) = \emptyset$ such that $C' \vdash S\, C$.

Proof. The full proof is given in Appendix A; here we only consider the crucial case where jdg' follows from jdg by an application of (gen). The situation is

$$\frac{jdg = (C \cup C_0, A \vdash e : t_0 \,\&\, b)}{jdg' = (C, A \vdash e : \forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0 \,\&\, b)}$$

where $\forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0$ is well-formed and where $\{\vec{\alpha}\vec{\beta}\} \cap FV(C, A, b) = \emptyset$ and where there exists S_0 with $Dom(S_0) \subseteq \{\vec{\alpha}\vec{\beta}\}$ such that $C \vdash S_0\, C_0$. Our assumptions are

$$C \cup C_0, A_n \vdash_n e_n : t_0 \,\&\, b_r \tag{1}$$
$$C \cup C_0 \vdash b_n \cup b_r \subseteq b \tag{2}$$

and we must show that there exists b'_r and S such that the following holds:

$$C, A_n \vdash_n e_n : \forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0 \,\&\, b'_r \tag{3}$$
$$C \vdash b_n \cup b'_r \subseteq b \tag{4}$$
$$Dom(S) \cap FV(A, b_n) = \emptyset \text{ and } C \vdash S\,(C \cup C_0).$$

We choose $b'_r = b_r$ and $S = S_0$ and then it will suffice to prove

$$\{\vec{\alpha}\vec{\beta}\} \cap FV(b_n, b_r) = \emptyset \tag{5}$$

for then (2) and Lemma 6(a) give that $C \cup S\, C_0 \vdash b_n \cup b_r \subseteq b$ which (by Lemma 7) implies (4); and since $FV(A_n) \setminus FV(A) \subseteq FV(b_n)$ holds by assumption we will be able to use (gen) to arrive at (3) from (1).

So we are left with the task of proving (5). By the assumption $FV(b) \cap \{\vec{\alpha}\vec{\beta}\} = \emptyset$ this can be done by showing

$$\forall \gamma' \in FV(b_n, b_r) : \exists \gamma \in FV(b) : C \cup C_0 \vdash \gamma' \leftarrow^* \gamma \tag{6}$$
$$\forall \gamma' \in \{\vec{\alpha}\vec{\beta}\} : (C \cup C_0 \vdash \gamma' \leftarrow^* \gamma \text{ implies } \gamma \in \{\vec{\alpha}\vec{\beta}\}). \tag{7}$$

(6) follows from (2) by Lemma 5, since $C \cup C_0$ is well-formed. (7) follows from repeated application of the below argument: if $C \cup C_0 \vdash \gamma' \leftarrow \gamma$ with $\gamma' \in \{\vec{\alpha}\vec{\beta}\}$ then by Fact 3 we are able to infer (as $\{\vec{\alpha}\vec{\beta}\} \cap FV(C) = \emptyset$) that $C_0 \vdash \gamma' \leftarrow \gamma$ and since $\forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0$ is well-formed (and thus $\{\vec{\alpha}\vec{\beta}\}^{C_0\uparrow} = \{\vec{\alpha}\vec{\beta}\}$) this implies $\gamma \in \{\vec{\alpha}\vec{\beta}\}$. □

Next some auxiliary results concerning the three kinds of concurrent transitions:

Lemma 38. Let C be well-formed and suppose that

$$C, A \vdash_s E[\texttt{channel ()}] : \sigma \,\&\, b.$$

Let ch be a channel identifier that does not occur in $E[\texttt{channel ()}]$; then there exists t_n and b_r such that

$$C \vdash \{t_n \text{ CHAN}\} \cup b_r \subseteq b \text{ and}$$
$$C, A[ch : t_n \texttt{ chan}] \vdash_n E[ch] : \sigma \,\&\, b_r.$$

Proof. The strongly normalised inference tree contains a judgement of form

$$C', A \vdash_s \texttt{channel () } : t' \,\&\, b'$$

where C' is well-formed (Fact 30). Since

$$\text{TypeOf}(\texttt{channel}) = \forall(\alpha\beta : \{\{\alpha \text{ CHAN}\} \subseteq \beta\}).\ \texttt{unit} \to^{\beta} (\alpha \texttt{ chan})$$

we can use Lemma 31 to find S such that

$$C' \vdash \{S\,\alpha \text{ CHAN}\} \subseteq S\,\beta \text{ and } C' \vdash S\,\alpha \texttt{ chan} \subseteq t' \text{ and } C' \vdash S\,\beta \subseteq b'.$$

Now define $t_n = S\,\alpha$ and $b_n = \{t_n \text{ CHAN}\}$, then

$$C', A[ch : t_n \texttt{ chan}] \vdash_n ch : t' \,\&\, \emptyset \text{ and } C' \vdash \{t_n \text{ CHAN}\} \cup \emptyset \subseteq b'$$

so as $FV(t_n) \subseteq FV(b_n)$ Lemma 37 gives us b_r such that

$$C, A[ch : t_n \texttt{ chan}] \vdash_n E[ch] : \sigma \,\&\, b_r \text{ and } C \vdash \{t_n \text{ CHAN}\} \cup b_r \subseteq b.$$

This completes the proof.

Lemma 39. Let C be well-formed and suppose that

$$C, A \vdash_s E[\texttt{fork}\,w] : \sigma \,\&\, b.$$

Then there exists b_r, t'', b'' such that

(a) $C, A \vdash_n E[\texttt{()}] : \sigma \,\&\, b_r$;
(b) $C, A \vdash_n w\ \texttt{()} : t'' \,\&\, b''$;
(c) $C \vdash b_r \subseteq b$.

Proof. The strongly normalised inference tree contains a judgement of form

$$C', A \vdash_s \texttt{fork}\, w : t' \,\&\, b'$$

where C' is well-formed (Fact 30). Since

$$\text{TypeOf}(\texttt{fork}) = \forall(\alpha\beta : \emptyset).\ (\texttt{unit} \to^{\beta} \alpha) \to^{\emptyset} \texttt{unit}$$

we from Lemma 31 get t_1, b_1 and S such that

$$C' \vdash \texttt{unit} \subseteq t' \text{ and } C' \vdash b_1 \subseteq b' \tag{8}$$

$$C', A \vdash_s w : t_1 \,\&\, b_1 \text{ and } C' \vdash t_1 \subseteq \texttt{unit} \to^{S\,\beta} S\,\alpha \tag{9}$$

and by Lemma 33 we infer that

$$C', A \vdash_n w : t_1 \,\&\, \emptyset. \tag{10}$$

From (8) we get

$$C', A \vdash_n () : t' \,\&\, \emptyset$$

and Lemma 37 (with $m = 0$ and $b_n = \emptyset$) then gives us a b_r such that

$$C, A \vdash_n E[()] : \sigma \,\&\, b_r \text{ and } C \vdash b_r \subseteq b$$

which yields the claims (a) and (c), and in addition an S' such that

$$Dom(S') \cap FV(A) = \emptyset \text{ and } C \vdash S'\, C'. \tag{11}$$

For the remaining claim (b), we from (9) and (10) infer that

$$C', A \vdash_n w\,() : S\,\alpha \,\&\, S\,\beta$$

so using (11) we (by Lemma 6 and Lemma 7) arrive at

$$C, A \vdash_n w\,() : t'' \,\&\, b''$$

for $t'' = S'\,S\,\alpha$ and $b'' = S'\,S\,\beta$, thus yielding the claim (b). □

Lemma 40. Let C be well-formed and let A be a channel environment and suppose that

$$C, A \vdash_s E_1[\texttt{sync}\,(\texttt{send}\,(\texttt{pair}\,ch\,w))] : \sigma_1 \,\&\, b_1 \tag{12}$$

and suppose that

$$C, A \vdash_s E_2[\texttt{sync}\,(\texttt{receive}\,ch)] : \sigma_2 \,\&\, b_2. \tag{13}$$

Let $A(ch) = t$ `chan`, then there exists b_s and b_r such that

(a) $C, A \vdash_n E_1[w] : \sigma_1 \,\&\, b_s$ and $C \vdash b_s \subseteq b_1$;
(b) $C, A \vdash_n w : t \,\&\, \emptyset$;
(c) $C, A \vdash_n E_2[w] : \sigma_2 \,\&\, b_r$ and $C \vdash b_r \subseteq b_2$.

Proof. The tree (12) will contain a judgement of form

$$C_1, A \vdash_s \texttt{sync}\,(\texttt{send}\,(\texttt{pair}\,ch\,w)) \,:\, t_1 \,\&\, b_1' \tag{14}$$

where C_1 is well-formed (Fact 30). Since

$$\text{TypeOf}(\texttt{sync}) = \forall(\alpha\beta : \emptyset).\ (\alpha\ \texttt{com}\ \beta) \to^{\beta}\ \alpha,$$

Lemma 31 together with Lemma 33 tells us that there exists t_3 and S_3 such that

$$C_1, A \vdash_n \texttt{send}\,(\texttt{pair}\,ch\,w) \,:\, t_3 \,\&\, \emptyset;$$

$$C_1 \vdash S_3\,\alpha \subseteq t_1;$$

$$C_1 \vdash t_3 \subseteq (S_3\,\alpha)\ \texttt{com}\ (S_3\,\beta).$$

As $\text{TypeOf}(\texttt{send}) = \forall(\alpha : \emptyset).\ (\alpha\ \texttt{chan}) \times \alpha \to^{\emptyset}\ (\alpha\ \texttt{com}\ \emptyset)$, Lemma 31 (together with Lemma 33) tells us that there exists t_4 and S_4 such that

$$C_1, A \vdash_n \texttt{pair}\,ch\,w \,:\, t_4 \,\&\, \emptyset;$$

$$C_1 \vdash (S_4\,\alpha)\ \texttt{com}\ \emptyset \subseteq t_3;$$

$$C_1 \vdash t_4 \subseteq (S_4\,\alpha)\ \texttt{chan} \times (S_4\,\alpha).$$

Since $\text{TypeOf}(\texttt{pair}) = \forall(\alpha_1\alpha_2 : \emptyset).\ \alpha_1 \to^{\emptyset}\ \alpha_2 \to^{\emptyset}\ \alpha_1 \times \alpha_2$, Lemma 31 (together with Lemma 33) tells us that there exists t_5, t_6, and S_5 such that

$$C_1, A \vdash_n ch \,:\, t_5 \,\&\, \emptyset; \tag{15}$$

$$C_1, A \vdash_n w \,:\, t_6 \,\&\, \emptyset; \tag{16}$$

$$C_1 \vdash S_5\,\alpha_1 \times S_5\,\alpha_2 \subseteq t_4;$$

$$C_1 \vdash t_5 \subseteq S_5\,\alpha_1 \text{ and } C_1 \vdash t_6 \subseteq S_5\,\alpha_2.$$

Since $A(ch) = t\ \texttt{chan}$ we infer from (15) that

$$C_1 \vdash t\ \texttt{chan} \subseteq t_5.$$

We now apply Lemma 4 repeatedly: from

$$C_1 \vdash (S_4\,\alpha)\ \texttt{com}\ \emptyset \subseteq t_3 \subseteq (S_3\,\alpha)\ \texttt{com}\ (S_3\,\beta)$$

$$C_1 \vdash t_5 \times t_6 \subseteq S_5\,\alpha_1 \times S_5\,\alpha_2 \subseteq t_4 \subseteq (S_4\,\alpha)\ \texttt{chan} \times (S_4\,\alpha)$$

we deduce that

$$C_1 \vdash S_4\,\alpha \subseteq S_3\,\alpha \subseteq t_1$$

$$C_1 \vdash t\ \texttt{chan} \subseteq t_5 \subseteq (S_4\,\alpha)\ \texttt{chan}$$

$$C_1 \vdash t_6 \subseteq S_4\,\alpha$$

From (16) we therefore get

$$C_1, A \vdash_n w \,:\, t_1 \,\&\, \emptyset$$

so by Lemma 37 applied to (12) and (14) (with $m = 0$ and $b_n = \emptyset$) we find b_s and S_1 such that

$C, A \vdash_n E_1[w] : \sigma_1 \,\&\, b_s$ and $C \vdash b_s \subseteq b_1$;

$Dom(S_1) \cap FV(A) = \emptyset$ and $C \vdash S_1\, C_1$. (17)

We have thus established (a). By exploiting the *contravariance*[2] of $\cdots$ `chan` (cf. the remarks concerning Figure 2), we get

$C_1 \vdash t_6 \subseteq S_4\, \alpha \subseteq t$ (18)

and from (16) therefore

$C_1, A \vdash_n w : t \,\&\, \emptyset$

which using (17) yields (as $FV(t) \subseteq FV(A)$)

$C, A \vdash_n w : t \,\&\, \emptyset$.

We have thus established (b).

Our remaining task is to show claim (c), where we first notice that the tree (13) will contain a judgement of form

$C_2, A \vdash_s \texttt{sync}\,(\texttt{receive}\, ch) : t_2 \,\&\, b_2'$ (19)

where C_2 is well-formed (Fact 30). Since

$\text{TypeOf}(\texttt{sync}) = \forall(\alpha\beta : \emptyset).\ (\alpha\ \texttt{com}\ \beta) \rightarrow^{\beta}\ \alpha$

Lemma 31 (together with Lemma 33) tells us that there exists t_7 and S_7 such that

$C_2, A \vdash_n \texttt{receive}\, ch : t_7 \,\&\, \emptyset$;

$C_2 \vdash S_7\, \alpha \subseteq t_2$;

$C_2 \vdash t_7 \subseteq (S_7\, \alpha)\ \texttt{com}\ (S_7\, \beta)$.

Since $\text{TypeOf}(\texttt{receive}) = \forall(\alpha : \emptyset).\ (\alpha\ \texttt{chan}) \rightarrow^{\emptyset}\ (\alpha\ \texttt{com}\ \emptyset)$, Lemma 31 tells us that there exists t_8 and S_8 such that

$C_2, A \vdash_n ch : t_8 \,\&\, \emptyset$; (20)

$C_2 \vdash (S_8\, \alpha)\ \texttt{com}\ \emptyset \subseteq t_7$;

$C_2 \vdash t_8 \subseteq (S_8\, \alpha)\ \texttt{chan}$.

Since $A(ch) = t\ \texttt{chan}$ we infer from (20) that

$C_2 \vdash t\ \texttt{chan} \subseteq t_8$.

We now apply Lemma 4 repeatedly: from

$C_2 \vdash (S_8\, \alpha)\ \texttt{com}\ \emptyset \subseteq t_7 \subseteq (S_7\, \alpha)\ \texttt{com}\ (S_7\, \beta)$

$C_2 \vdash t\ \texttt{chan} \subseteq t_8 \subseteq (S_8\, \alpha)\ \texttt{chan}$

[2] For later reference we note that if we were to use also the covariance of $\cdots$ `chan` we would additionally get that $C_1 \vdash t \subseteq S_4\, \alpha \subseteq t_1$.

we get, by exploiting the *covariance* of $\cdots$ `chan` (cf. the remarks concerning Figure 2),

$$C_2 \vdash t \subseteq S_8\,\alpha \subseteq S_7\,\alpha \subseteq t_2. \tag{21}$$

As the inference (13) is constraint saturated we have $C \subseteq C_2$ so by Lemma 7 we can deduce from claim (b) that

$$C_2, A \vdash_n w : t \,\&\, \emptyset$$

so by applying (sub) we arrive at

$$C_2, A \vdash_n w : t_2 \,\&\, \emptyset.$$

By applying Lemma 37 on (13) and (19) (with $m = 0$ and $b_n = \emptyset$) we find b_r such that

$$C, A \vdash_n E_2[w] : \sigma_2 \,\&\, b_r \text{ and } C \vdash b_r \subseteq b_2$$

which establishes (c) thus completing the proof. □

We are now able to formulate what it means for our system to be semantically sound. We write $C, A \vdash PP : PT \,\&\, PB$, where PT (respectively PB) is a mapping from process identifiers into types (respectively behaviours), if the domains of PP, PT and PB are equal and if for all $p \in Dom(PP)$ we have $C, A \vdash PP(p) : PT(p) \,\&\, PB(p)$.

Theorem 41. *Semantic (concurrent) soundness*

Let C be well-formed, let A be a channel environment, and suppose

$$C, A \vdash_n PP : PT \,\&\, PB.$$

If $PP \xrightarrow{a} PP'$ there exists PT', PB' and channel environment A' such that

$$C, A' \vdash_n PP' : PT' \,\&\, PB'$$

and such that if p is in the domain of PP then $PT'(p) = PT(p)$ and such that if ch occurs in PP then $A'(ch) = A(ch)$.

Furthermore we have the following property:

- If $a = \texttt{seq}$ then $PB'(p) = PB(p)$ for all p in the domain of PP.
- If $a = p_0$ `chan` ch then there exists t_0 such that $A'(ch) = t_0$ `chan`, such that
  $C \vdash \{t_0 \text{ CHAN}\} \cup PB'(p_0) \subseteq PB(p_0)$
  and such that $PB'(p) = PB(p)$ for $p \in Dom(PP) \setminus \{p_0\}$.
- If $a = p_0$ `fork` p' then
  $C \vdash PB'(p_0) \subseteq PB(p_0)$
  and $PB'(p) = PB(p)$ for $p \in Dom(PP) \setminus \{p_0\}$.
- If $a = p_1$ `comm` p_2 then
  $C \vdash PB'(p_1) \subseteq PB(p_1)$ and $C \vdash PB'(p_2) \subseteq PB(p_2)$
  and $PB'(p) = PB(p)$ for $p \in Dom(PP) \setminus \{p_1, p_2\}$.

Proof. By Fact 13 we can assume the inference trees in $C, A \vdash_n PP : PT \,\&\, PB$ to be strongly normalised. We perform case analysis on the action label a:

$a = \texttt{seq}$**:** It follows from Theorem 36 that we can use $PT' = PT$, $PB' = PB$ and $A' = A$.

$a = p_0$ `chan` ch**:** It follows from Lemma 38 that there exists t_0 and b_r such that the claim follows with $PT' = PT$, $PB' = PB[p_0 : b_r]$ and $A' = A[ch : t_0\ \texttt{chan}]$. (For $p \neq p_0$ ($\in Dom(PP)$) we must show that $C, A \vdash_n PP(p) : PT(p) \,\&\, PB(p)$ implies $C, A' \vdash_n PP(p) : PT(p) \,\&\, PB(p)$, but this follows from Fact 9.)

$a = p_0$ `fork` p'**:** It follows from Lemma 39 that there exists t'', b'' and b_r such that we can use $PT' = PT[p' : t'']$, $PB' = PB[p_0 : b_r][p' : b'']$ and $A' = A$.

$a = p_1$ `comm` p_2**:** It follows from Lemma 40 that there exists b_s and b_r such that we can use $PT' = PT$, $PB' = PB[p_1 : b_s][p_2 : b_r]$ and $A' = A$. □

Remark. Theorem 41 makes it explicit that the type of a channel does not change after it has been allocated. This should be compared with the subject reduction result in [11, lemma 5.2], the formulation of which allows one the possibility of assigning different types to the same location at various stages (although apparently it is always possible to choose the same type and still get subject reduction).

Remark. Theorem 41 says that if we start with a correctly typed program then we are never going to encounter programs that are not correctly typed. One consequence of this is that Lemma 40 will be applicable at all stages; this is a result that ensures that the value sent can always be given the type allowed on the channel on which it was sent, that having sent the value we still have a correctly typed sender, and that having received the value we still have a correctly typed receiver. However, the statement of Lemma 40 does not directly relate:

- the type t_6 of the value w actually communicated (see line 16),
- the type t of the entities allowed to be communicated over the channel,
- the type t_1 that the sender thinks was communicated (see line 14), and
- the type t_2 that the receiver thinks was communicated (see line 19).

However, by inspecting the proof of Lemma 40 one may note that the following relations are established (by (18), footnote 2 and (21)):

$$C_1 \vdash t_6 \subseteq t \qquad\qquad C_1 \vdash t \subseteq t_1 \qquad\qquad C_2 \vdash t \subseteq t_2$$

Here the constraint sets C_1 and C_2 are those corresponding to the point of sending and receiving, respectively. Thus we can be ensured that a value is always received with a type that is larger than the type it actually had when communicated. (It is possible for the sender to think that an even larger type was communicated, but this causes no harm.)

4 Conclusion

We have given a formal justification of the semantic soundness of a previously developed annotated type and effect system that integrates polymorphism, subtyping and effects [6]; in particular it was highlighted that the judicious choice of generalisation strategy in the system plays a crucial role, as witnessed by the proof of Lemma 37. Although the development was performed for a fragment of Concurrent ML we believe it equally possible for Standard ML with references; moreover we conjecture that the development can be extended to incorporate causality in the behaviours.

Acknowledgement. This work has been supported in part by the *DART* project (Danish Natural Science Research Council) and the *LOMAPS* project (ESPRIT BRA project 8130); it represents joint work among the authors. We would like to thank the anonymous referees for their thought-provoking comments on the preliminary version of this work.

References

1. M. Debbabi and D. Bolignano: A semantic theory for ML higher-order concurrency primitives. In *ML with Concurrency: Design, Analysis, Implementation and Application* (editor: Flemming Nielson), Springer-Verlag, 1996.
2. M. Felleisen and D.P. Friedman: Control operators, the SECD-Machine, and the λ-calculus. *Formal Descriptions of Programming Concepts III*, North-Holland, 1986.
3. X. Leroy and P. Weis: Polymorphic type inference and assignment. In *Proc. POPL '91*, pages 291–302. ACM Press, 1991.
4. R. Milner: A theory of type polymorphism in programming. *Journal of Computer and System Sciences*, 17:348–375, 1978.
5. H.R. Nielson and F. Nielson: Higher-order concurrent programs with finite communication topology. In *Proc. POPL'94*, pages 84–97. ACM Press, 1994.
6. H.R. Nielson, F. Nielson, T. Amtoft: Polymorphic subtypes for effect analysis: the static semantics. This volume of SLNCS, 1997.
7. G.D. Plotkin: A structural approach to operational semantics. Report DAIMI FN-19, Aarhus University, Denmark, 1981.
8. J. H. Reppy: Concurrent ML: Design, application and semantics. In *Proc. Functional Programming, Concurrency, Simulation and Automated Reasoning*, pages 165–198. SLNCS 693, 1993.
9. J. P. Talpin and P. Jouvelot: The type and effect discipline. *Information and Computation*, 111, 1994.
10. M. Tofte: Type inference for polymorphic references. *Information and Computation*, 89:1–34, 1990.
11. A.K. Wright and M. Felleisen: A syntactic approach to type soundness. *Information and Computation*, 115, pages 38–94, 1994.

A Proofs of Main Results

Reasoning about proof trees

Fact 27 Given $jdg' = C', A' \vdash E[e] : \sigma' \,\&\, b'$; then there exists (at least one) judgement jdg of form $C, A \vdash e : \sigma \,\&\, b$ such that jdg occurs at E in the inference tree for jdg'. If jdg' is (strongly) normalised we can assume that jdg is (strongly) normalised.

Proof. The proof is by induction in the inference tree for jdg'. If $E = [\,]$ we can use $jdg = jdg'$, so assume $E \neq [\,]$. Hence the last rule applied in the inference tree for jdg' is none of the following: (con), (id), (abs), or (rec). If (sub), (ins) or (gen) has been applied the induction hypothesis clearly yields the claim; notice that if jdg' is normalised then it cannot be the case that (ins) has been applied.

So we are left with (app), (let) and (if); we only consider (app) as the other cases are similar. Then E takes either the form $E_1\, e_2$ or the form $w_1\, E_2$; we consider the former only as the latter is similar.

The situation thus is that $E[e] = E_1[e]\, e_2$ so the left premise of jdg' is of form $C'', A'' \vdash E_1[e] : \sigma'' \,\&\, b''$ (abbreviated jdg''). Inductively we can assume that there exists jdg which occurs at E_1 in the inference tree for jdg''; but this shows that jdg occurs at E in the inference tree for jdg'. □

Lemma 31 Suppose that C is well-formed and that

$$C, A \vdash_n \; c\, e_1 \cdots e_n \; : \; t \,\&\, b \;\; (n \geq 0)$$

and that TypeOf(c) is of form

$$\forall(\vec{\alpha}\vec{\beta} : C_0).\; t'_1 \to^{b'_1} \; \cdots t'_m \to^{b'_m} \; t''$$

and that if c is a base function then $m \geq n$.

Then in all cases (i.e. also if c is a constructor) we can write

$$\text{TypeOf}(c) = \forall(\vec{\alpha}\vec{\beta} : C_0).\; t'_1 \to^{b'_1} \; \cdots t'_n \to^{b'_n} \; t' \tag{1}$$

and there exists S, $t_1 \cdots t_n$, and $b_1 \cdots b_n$, such that

$Dom(S) \subseteq \{\vec{\alpha}\vec{\beta}\}$ and $C \vdash S\, C_0$ and $C \vdash S\, t' \subseteq t$;

for all $i \in \{1 \cdots n\}$: $C, A \vdash_s \; e_i \; : \; t_i \,\&\, b_i$ and $C \vdash t_i \subseteq S\, t'_i$ and $C \vdash b_i \subseteq b$ and $C \vdash S\, b'_i \subseteq b$.

Similarly, if TypeOf(c) $= t'_1 \to^{b'_1} \; \cdots t'_m \to^{b'_m} \; t''$ in which case $\{\vec{\alpha}\vec{\beta}\} = \emptyset$ and $C_0 = \emptyset$ (so we have $S = \text{Id}$).

Proof. By Fact 13 we can assume that we in fact have $C, A \vdash_s \; c\, e_1 \cdots e_n \; : \; t \,\&\, b$. We perform induction in n. If $n = 0$ we can trivially always assume (1), i.e. that TypeOf(c) takes the form $\forall(\vec{\alpha}\vec{\beta} : C_0).\; t'$, and the claim is that if $C, A \vdash_s \; c \; : \; t \,\&\, b$ then there exists S with $Dom(S) \subseteq \{\vec{\alpha}\vec{\beta}\}$ and $C \vdash S\, C_0$ such that $C \vdash S\, t' \subseteq t$.

But since $C, A \vdash_s\ c : t \,\&\, b$ is constructed by an application of (con) followed by an application of (ins) followed by zero or more applications of (sub), this is immediate.

Next consider the inductive step. The situation is that there exists t_n, t^-, b_n, b' and b'' such that

$$\frac{C, A \vdash_s\ c\, e_1 \cdots e_{n-1} : t_n \to^{b''}\ t^- \,\&\, b' \qquad C, A \vdash_s\ e_n : t_n \,\&\, b_n}{C, A \vdash_s\ c\, e_1 \cdots e_n : t \,\&\, b}\ \text{(app)(sub)}^*$$

where $C \vdash t^- \subseteq t$ and $C \vdash b' \cup b_n \cup b'' \subseteq b$.

By the induction hypothesis we infer that in all cases it holds that

TypeOf(c) takes the form $\forall(\vec{\alpha}\vec{\beta} : C_0).\ t'_1 \to^{b'_1}\ \cdots t'_{n-1} \to^{b'_{n-1}}\ t'''$

and that there exists S, $t_1 \cdots t_{n-1}$, and $b_1 \cdots b_{n-1}$, such that

$$\begin{array}{l} \mathit{Dom}(S) \subseteq \{\vec{\alpha}\vec{\beta}\} \text{ and } C \vdash S\, C_0; \\ C \vdash S\, t''' \subseteq t_n \to^{b''}\ t^-; \qquad (2) \\ \text{for all } i \in \{1 \cdots n-1\}\text{: } C, A \vdash_s\ e_i : t_i \,\&\, b_i \text{ and } C \vdash t_i \subseteq S\, t'_i \text{ and} \\ C \vdash b_i \subseteq b' \subseteq b \text{ and } C \vdash S\, b'_i \subseteq b' \subseteq b. \end{array}$$

Since C is well-formed we can apply Lemma 4 on (2) to infer that $S\, t'''$ is a function type. If c is a constructor Fact 1 tells us that t''' cannot be a variable; hence in all cases we can write $t''' = t'_n \to^{b'_n}\ t'$ which amounts to (1). Lemma 4 further tells us that $C \vdash t_n \subseteq S\, t'_n$ and that $C \vdash S\, b'_n \subseteq b'' \subseteq b$ and that $C \vdash S\, t' \subseteq t^- \subseteq t$. Thus all our proof obligations are fulfilled. □

Lemma 32 Suppose that $C, A[x : \sigma'] \vdash_n\ e : \sigma \,\&\, b$ and $C, A \vdash_n\ e' : \sigma' \,\&\, \emptyset$; then $C, A \vdash_n\ e[e'/x] : \sigma \,\&\, b$.

Proof. Induction in the shape of the proof tree for $C, A[x : \sigma'] \vdash_n\ e : \sigma \,\&\, b$; we perform case analysis on the way it is constructed (cf. Definition 10).

(con) or (con)(ins) has been applied: Then e is a constant, and $e[e'/x] = e$ so the claim is clear.

(id) or (id)(ins) has been applied: Then e is an identifier y. If $y \neq x$ then $e[e'/x] = e$ and the claim is clear since $A[x : \sigma'](y) = A(y)$.

If $y = x$ then the inference takes the form

$$\frac{C, A[x : \sigma'] \vdash x\ :\ \sigma' \,\&\, \emptyset}{C, A[x : \sigma'] \vdash x\ :\ t \,\&\, \emptyset}$$

where the last rule follows by zero or one application of (ins). We must show

$$C, A \vdash_n\ e' : t \,\&\, \emptyset$$

but this follows from the second part of the assumption, using Lemma 11.

(abs) has been applied: Here the inference takes the form

$$\frac{C, A[x : \sigma'][y : t_1] \vdash_n e : t_2 \,\&\, b}{C, A[x : \sigma'] \vdash_n \texttt{fn}\ y \Rightarrow e : t_1 \to^b t_2 \,\&\, \emptyset}$$

where we can assume (by suitable alpha-renaming) that $y \neq x$ and that y does not occur in e'. Hence we can apply Fact 8 and Fact 9 to get

$C, A[y : t_1][x : \sigma'] \vdash_n e : t_2 \,\&\, b$ with the same shape as the premise
$C, A[y : t_1] \vdash_n e' : \sigma' \,\&\, \emptyset$.

We can thus apply the induction hypothesis and subsequently use (abs) to construct an inference tree whose last inference is

$$\frac{C, A[y : t_1] \vdash_n e[e'/x] : t_2 \,\&\, b}{C, A \vdash_n \texttt{fn}\ y \Rightarrow e[e'/x] : t_1 \to^b t_2 \,\&\, \emptyset}$$

which is as desired since $(\texttt{fn}\ y \Rightarrow e)[e'/x] = (\texttt{fn}\ y \Rightarrow e[e'/x])$.

(app) has been applied: With $C = C_1 \cup C_2$ the inference takes the form

$$\frac{C_1, A[x : \sigma'] \vdash_n e_1 : t_2 \to^b t_1 \,\&\, b_1 \qquad C_2, A[x : \sigma'] \vdash_n e_2 : t_2 \,\&\, b_2}{C, A[x : \sigma'] \vdash_n e_1\, e_2 : t_1 \,\&\, (b_1 \cup b_2 \cup b)}$$

and by Lemma 7 we infer that

$$C, A[x : \sigma'] \vdash_n e_1 : t_2 \to^b t_1 \,\&\, b_1 \text{ and } C, A[x : \sigma'] \vdash_n e_2 : t_2 \,\&\, b_2$$

with the same shape as the premises; so we can apply the induction hypothesis twice and subsequently use (app) to construct an inference tree whose last inference is

$$\frac{C, A \vdash_n e_1[e'/x] : t_2 \to^b t_1 \,\&\, b_1 \qquad C, A \vdash_n e_2[e'/x] : t_2 \,\&\, b_2}{C, A \vdash_n e_1[e'/x]\, e_2[e'/x] : t_1 \,\&\, (b_1 \cup b_2 \cup b)}$$

which is as desired since $(e_1\, e_2)[e'/x] = e_1[e'/x]\, e_2[e'/x]$.

(let), (rec) or (if) has been applied: Similar to the above two cases, exploiting Fact 8 and Fact 9 and we only spell the case (rec) out in detail. Here the inference takes the form

$$\frac{C, A[x : \sigma'][f : t] \vdash_n \texttt{fn}\ y \Rightarrow e : t \,\&\, b}{C, A[x : \sigma'] \vdash_n \texttt{rec}\ f\ y \Rightarrow e : t \,\&\, b}$$

where we can assume that $y \neq x$, $f \neq x$ and that neither y nor f occurs in e'. Hence we can apply Fact 8 and Fact 9 to get

$C, A[f : t][x : \sigma'] \vdash_n \texttt{fn}\ y \Rightarrow e : t \,\&\, b$
$C, A[f : t] \vdash_n e' : \sigma' \,\&\, \emptyset$

and as the former inference has the same shape as the premise we can apply the induction hypothesis to infer

$C, A[f : t] \vdash_n (\texttt{fn}\ y \Rightarrow e)[e'/x] : t \,\&\, b$

which since $y \neq x$ and y is not free in e' amounts to

$C, A[f : t] \vdash_n \texttt{fn}\ y \Rightarrow e[e'/x] : t \,\&\, b.$

By applying (rec) we get

$C, A \vdash_n \texttt{rec}\ f\ y \Rightarrow e[e'/x] : t \,\&\, b$

which is as desired since $(\texttt{rec}\ f\ y \Rightarrow e)[e'/x] = (\texttt{rec}\ f\ y \Rightarrow e[e'/x])$.

(sub) has been applied: Here the inference takes the form

$$\frac{C, A[x : \sigma'] \vdash_n e : t \,\&\, b}{C, A[x : \sigma'] \vdash_n e : t' \,\&\, b'}$$

with $C \vdash t \subseteq t'$ and $C \vdash b \subseteq b'$ so we can apply the induction hypothesis and subsequently use (sub) to construct an inference tree whose last inference is

$$\frac{C, A \vdash_n e[e'/x] : t \,\&\, b}{C, A \vdash_n e[e'/x] : t' \,\&\, b'}$$

(gen) has been applied: Here the inference takes the form

$$\frac{C \cup C_0, A[x : \sigma'] \vdash_n e : t_0 \,\&\, b}{C, A[x : \sigma'] \vdash_n e : ts \,\&\, b}$$

where $ts = \forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0$ is well-formed, solvable from C, and satisfies $\{\vec{\alpha}\vec{\beta}\} \cap FV(C, A[x : \sigma'], b) = \emptyset$. By Lemma 7 we have

$C \cup C_0, A \vdash_n e' : \sigma' \,\&\, \emptyset$

so we can apply the induction hypothesis to get

$C \cup C_0, A \vdash_n e[e'/x] : t_0 \,\&\, b.$

We can then apply (gen) (since $\{\vec{\alpha}\vec{\beta}\} \cap FV(C, A, b) = \emptyset$) to arrive at the desired judgement $C, A \vdash_n e[e'/x] : ts \,\&\, b$. □

Lemma 33 Suppose that $C, A \vdash_n w : \sigma \,\&\, b$ with C well-formed; then

$C, A \vdash_n w : \sigma \,\&\, \emptyset.$

Proof. It is enough to consider the case where σ is a type t, for if the inference

$$\frac{C \cup C_0, A \vdash w : t_0 \,\&\, b}{C, A \vdash w : \forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0 \,\&\, b}\ (\text{gen})$$

is valid it remains valid when b is replaced by $\emptyset$. We now prove the claim by induction in the size of w, and the only interesting case is where $w = c\, w_1 \cdots w_n$ for $n \geq 1$ and with c being a constructor.

Lemma 31 combined with Fact 1 tells us that

TypeOf(c) takes the form $\forall(\vec{\alpha}\vec{\beta} : C_0).\ t'_1 \to^{\emptyset} \cdots t'_n \to^{\emptyset} t'$

and Lemma 31 further tells us that there exists $t_1 \cdots t_n$, $b_1 \cdots b_n$, and S with $Dom(S) \subseteq \{\vec{\alpha}\vec{\beta}\}$ and $C \vdash S\, C_0$ such that

$C \vdash S\, t' \subseteq t$;

for all $i \in \{1 \cdots n\}$: $C, A \vdash_s w_i : t_i \,\&\, b_i$ and $C \vdash t_i \subseteq S\, t'_i$.

The induction hypothesis tells us that

for all $i \in \{1 \cdots n\}$: $C, A \vdash_n w_i : t_i \,\&\, \emptyset$

and by using (con), (ins) and (sub) we have

$C, A \vdash_n c : t_1 \to^{\emptyset} \cdots t_n \to^{\emptyset} t \,\&\, \emptyset$

so we can clearly construct the desired judgement $C, A \vdash_n c\, w_1 \cdots w_n : t \,\&\, \emptyset$. □

Concurrent soundness

Lemma 37 Suppose the judgement $jdg = (C, A \vdash e : \sigma \,\&\, b)$ occurs at E with depth n' in the strongly normalised inference $jdg' = (C', A \vdash_s E[e] : \sigma' \,\&\, b')$ where C' (and by Fact 30 then also C) is well-formed.

Let b_n be a behaviour and let A_n be of form $A[x_1 : \sigma_1][\cdots][x_m : \sigma_m]$ with $m \geq 0$, such that $x_1 \cdots x_m$ do not occur in $E[e]$ and such that $FV(\sigma_1) \cup \cdots \cup FV(\sigma_m) \subseteq FV(b_n)$.

Let e_n be an expression and b_r a behaviour such that

$C, A_n \vdash_n e_n : \sigma \,\&\, b_r$ and
$C \vdash b_n \cup b_r \subseteq b$.

Then there exists b'_r such that

$C', A_n \vdash_n E[e_n] : \sigma' \,\&\, b'_r$ and
$C' \vdash b_n \cup b'_r \subseteq b'$.

Moreover, there exists S with $Dom(S) \cap FV(A, b_n) = \emptyset$ such that $C' \vdash S\, C$.

Proof. We perform induction in n': if $n' = 0$ then $E = [\,]$, $C' = C$, $\sigma' = \sigma$, $b' = b$ and the claim is trivial as we can choose $b'_r = b_r$ and $S = \mathrm{Id}$.

If $n' > 1$ then by Fact 28 there exists evaluation contexts E_1 and E_2 with $E = E_2[E_1]$ and judgement $jdg'' = (C'', A \vdash_s e'' : \sigma'' \,\&\, b'')$ such that

jdg occurs at E_1 with depth $< n'$ in the inference tree for jdg''; and
jdg'' occurs at E_2 with depth $< n'$ in the inference tree for jdg'.

C'' is well-formed by Fact 30, so if $C, A_n \vdash_n e_n : \sigma \,\&\, b_r$ and $C \vdash b_n \cup b_r \subseteq b$ we can apply the induction hypothesis (with jdg and jdg'') to infer that there exists b_r'' and S_1 such that $C'', A_n \vdash_n E_1[e_n] : \sigma'' \,\&\, b_r''$ and $C'' \vdash b_n \cup b_r'' \subseteq b''$ and $Dom(S_1) \cap FV(A, b_n) = \emptyset$ and $C'' \vdash S_1\, C$. We can then apply the induction hypothesis once more (with jdg'' and jdg') to infer that there exists b_r' and S_2 such that $C', A_n \vdash_n E_2[E_1[e_n]] : \sigma' \,\&\, b_r'$ and $C' \vdash b_n \cup b_r' \subseteq b'$ and $Dom(S_2) \cap FV(A, b_n) = \emptyset$ and $C' \vdash S_2\, C''$. This is as desired, since with $S = S_2\, S_1$ we have $Dom(S) \cap FV(A, b_n) = \emptyset$ and (by Lemma 6 and 7) $C' \vdash S\, C$.

So we are left with the case $n' = 1$. We perform case analysis on E:

$\underline{E = E_1\, e_2}$**:** Here $E_1 = [\,]$ and the situation is:

$$\frac{jdg = (C, A \vdash_n e_1 : (t_2 \to^{b_0} t_1) \,\&\, b_1) \qquad C, A \vdash_n e_2 : t_2 \,\&\, b_2}{jdg' = (C, A \vdash e_1\, e_2 : t_1 \,\&\, (b_1 \cup b_2 \cup b_0))}$$

and our assumptions are

$C, A_n \vdash_n e_n : t_2 \to^{b_0} t_1 \,\&\, b_r$ and
$C \vdash b_n \cup b_r \subseteq b_1$

and we must show that there exists b_r' and S such that

$$\begin{array}{l} C, A_n \vdash_n e_n\, e_2 : t_1 \,\&\, b_r' \\ C \vdash b_n \cup b_r' \subseteq b_1 \cup b_2 \cup b_0 \\ Dom(S) \cap FV(A, b_n) = \emptyset \text{ and } C \vdash S\, C. \end{array} \tag{3}$$

We can choose $b_r' = b_r \cup b_2 \cup b_0$ and $S = \mathrm{Id}$: then the only non-trivial claim is (3) which will follow provided we can show that

$C, A_n \vdash_n e_2 : t_2 \,\&\, b_2$

but this follows from Fact 9 since $x_1 \cdots x_m$ do not occur in e_2.

$\underline{E = w\, E_2}$**:** Here $E_2 = [\,]$ and the situation is:

$$\frac{C, A \vdash_n w : (t_2 \to^{b_0} t_1) \,\&\, b_1 \qquad jdg = (C, A \vdash_n e_2 : t_2 \,\&\, b_2)}{jdg' = (C, A \vdash w\, e_2 : t_1 \,\&\, (b_1 \cup b_2 \cup b_0))}$$

and our assumptions are

$C, A_n \vdash_n e_n : t_2 \,\&\, b_r$ and
$C \vdash b_n \cup b_r \subseteq b_2$

and we must show that there exists b_r' and S such that

$C, A_n \vdash_n w\, e_n : t_1 \,\&\, b_r'$ and
$C \vdash b_n \cup b_r' \subseteq b_1 \cup b_2 \cup b_0$ and
$Dom(S) \cap FV(A, b_n) = \emptyset$ and $C \vdash S\, C$.

By Lemma 33 and Fact 9 we infer that

$C, A_n \vdash_n w : (t_2 \to^{b_0} t_1) \,\&\, \emptyset$

which shows than we can use $b_r' = b_r \cup b_0$ and trivially $S = \mathrm{Id}$.

$\underline{E = \texttt{let}\ x = E_1\ \texttt{in}\ e_2\texttt{:}}$ Here $E_1 = [\,]$ and the situation is:

$$\frac{jdg = (C, A \vdash_n\ e_1\ :\ ts_1 \,\&\, b_1) \qquad C, A[x : ts_1] \vdash_n\ e_2\ :\ t_2 \,\&\, b_2}{jdg' = (C, A \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2\ :\ t_2 \,\&\, (b_1 \cup b_2))}$$

and our assumptions are

$C, A_n \vdash_n\ e_n\ :\ ts_1 \,\&\, b_r$ and
$C \vdash b_n \cup b_r \subseteq b_1$

and we must show that there exists b'_r and S such that

$$\begin{array}{l} C, A_n \vdash_n\ \texttt{let}\ x = e_n\ \texttt{in}\ e_2\ :\ t_2 \,\&\, b'_r \\ C \vdash b_n \cup b'_r \subseteq b_1 \cup b_2 \\ Dom(S) \cap FV(A, b_n) = \emptyset \text{ and } C \vdash S\,C. \end{array} \tag{4}$$

We can choose $b'_r = b_r \cup b_2$ and $S = \mathrm{Id}$: then the only non-trivial claim is (4) which will follow provided we can show that

$$C, A_n[x : ts_1] \vdash_n\ e_2\ :\ t_2 \,\&\, b_2.$$

But this follows from Fact 9 and Fact 8 since all of $x_1 \cdots x_m$ are $\neq x$ and do not occur in e_2.

$\underline{E = \texttt{if}\ E_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2\texttt{:}}$ Here $E_0 = [\,]$ and the situation is:

$$\frac{jdg = (C, A \vdash_n\ e_0\ :\ \texttt{bool} \,\&\, b_0) \qquad C, A \vdash_n\ e_1\ :\ t \,\&\, b_1 \qquad C, A \vdash_n\ e_2\ :\ t \,\&\, b_2}{jdg' = (C, A \vdash \texttt{if}\ e_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2\ :\ t \,\&\, b_0 \cup (b_1 \cup b_2))}$$

and our assumptions are

$C, A_n \vdash_n\ e_n\ :\ \texttt{bool} \,\&\, b_r$ and
$C \vdash b_n \cup b_r \subseteq b_0$

and we must show that there exists b'_r and S such that

$C, A_n \vdash_n\ \texttt{if}\ e_n\ \texttt{then}\ e_1\ \texttt{else}\ e_2\ :\ t \,\&\, b'_r$ and
$C \vdash b_n \cup b'_r \subseteq b_0 \cup (b_1 \cup b_2)$ and
$Dom(S) \cap FV(A, b_n) = \emptyset$ and $C \vdash S\,C$.

We can choose $b'_r = b_r \cup (b_1 \cup b_2)$ and $S = \mathrm{Id}$; then the claims will follow since by Fact 9 we have

$$C, A_n \vdash_n\ e_1\ :\ t \,\&\, b_1 \text{ and } C, A_n \vdash_n\ e_2\ :\ t \,\&\, b_2.$$

$\underline{E = [\,]}$**:** In this case jdg' follows from jdg by one application of either (sub), (ins) or (gen).

(sub) has been applied: the situation is

$$\frac{jdg = (C, A \vdash_n e : t \,\&\, b)}{jdg' = (C, A \vdash e : t' \,\&\, b')}$$

where $C \vdash t \subseteq t'$ and $C \vdash b \subseteq b'$. Our assumptions are

$C, A_n \vdash_n e_n : t \,\&\, b_r$ and
$C \vdash b_n \cup b_r \subseteq b$.

and we must show that there exists b'_r and S such that

$C, A_n \vdash_n e_n : t' \,\&\, b'_r$ and
$C \vdash b_n \cup b'_r \subseteq b'$ and
$Dom(S) \cap FV(A, b_n) = \emptyset$ and $C \vdash S\,C$.

But we can clearly choose $b'_r = b_r$ and $S = \mathrm{Id}$.

(ins) has been applied: the situation is

$$\frac{jdg = (C, A \vdash e : \forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0 \,\&\, b)}{jdg' = (C, A \vdash e : S_0\, t_0 \,\&\, b)}$$

where $\forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0$ is solvable from C by S_0 (and where the premise is constructed by (con) or (id)). Our assumptions are

$C, A_n \vdash_n e_n : \forall(\vec{\alpha}\vec{\beta} : C_0).\ t_0 \,\&\, b_r$ and
$C \vdash b_n \cup b_r \subseteq b$.

and we must show that there exists b'_r and S such that

$C, A_n \vdash_n e_n : S_0\, t_0 \,\&\, b'_r$ and
$C \vdash b_n \cup b'_r \subseteq b$ and
$Dom(S) \cap FV(A, b_n) = \emptyset$ and $C \vdash S\,C$.

But we can clearly choose $b'_r = b_r$ and $S = \mathrm{Id}$, using Lemma 11.

(gen) has been applied: this case has been covered in the main text. □