Polymorphic Subtyping for Effect Analysis: the Semantics

T.Amtoft & F.Nielson & H.R.Nielson & J.Ammann

Computer Science Department, Aarhus University, Denmark

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

April 17, 1996

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 for Concurrent ML 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 [3] we developed an annotated type and effect system for a fragment of Concurrent ML. This system allowed the integration of ML-style polymorphism (the `let`-construct), subtyping (with the usual contravariant ordering for function space), and effects (for the set of "dangerous variables"). One key idea in the design of the annotated type and effect system was the following [3]:

- Carefully taking effects into account when deciding the set of variables over which to generalise in the rule for `let` in the inference system; this involves taking upwards closure with respect to a constraint set and is essential for maintaining semantic soundness and a number of substitution properties.

This is highlighted in the present paper. First we define a small step operational semantics [4] for Concurrent ML. It employs one system for the sequential

components and another for the concurrent components and as in [5, 2] we use evaluation contexts [1]. Next we extend the repertoire of techniques [3] for normalising and manipulating the inference trees of the annotated type and effect system. Finally, we show that the system is indeed semantically sound with respect to the operational semantics.

2 Inference System and Semantics

We first briefly recapitulate the inference system presented in [3]. Expressions and constants are given by

$$\begin{array}{rcl} e & ::= & c \mid x \mid \texttt{fn}\ x \Rightarrow e \mid e_1\, e_2 \mid \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \\ & \mid & \texttt{rec}\ f\ x \Rightarrow e \mid \texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 \\ c & ::= & \texttt{()} \mid \texttt{true} \mid \texttt{false} \mid n \mid \texttt{+} \mid \texttt{*} \mid \texttt{=} \mid \cdots \\ & \mid & \texttt{pair} \mid \texttt{fst} \mid \texttt{snd} \mid \texttt{nil} \mid \texttt{cons} \mid \texttt{hd} \mid \texttt{tl} \mid \texttt{isnil} \\ & \mid & \texttt{send} \mid \texttt{receive} \mid \texttt{sync} \mid \texttt{channel} \mid \texttt{fork} \end{array}$$

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`, and the non-sequential base functions `sync`, `channel` and `fork`.

Types and behaviours are given by

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

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.

Fact 2.1 Let c be a constructor. Then there exists $t'_1,\cdots,t'_m$ ($m \geq 0$) and t' such that

$$\text{TypeOf(c)} = \forall(\vec{\alpha}\vec{\beta} : \emptyset).\ t'_1 \rightarrow^{\emptyset} \cdots t'_m \rightarrow^{\emptyset} t'$$

where t' is not a function type (i.e. the decomposition is "maximal") nor a type variable.

The ordering among types and behaviours is depicted in Figure 2; in particular notice that the ordering is contravariant in the argument position of a function type and that both $t\ \texttt{chan} \subseteq t'\ \texttt{chan}$ and $\{t\ \textsc{chan}\} \subseteq \{t'\ \textsc{chan}\}$ demand that

c	TypeOf(c)
`+`	$\texttt{int} \times \texttt{int} \rightarrow^{\emptyset} \texttt{int}$
`pair`	$\forall(\alpha_1\alpha_2 : \emptyset).\ \alpha_1 \rightarrow^{\emptyset} \alpha_2 \rightarrow^{\emptyset} \alpha_1 \times \alpha_2$
`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`	$\forall(\alpha\beta : \{\{\alpha\ \textsc{chan}\} \subseteq \beta\}).\ \texttt{unit} \rightarrow^{\beta} (\alpha\ \texttt{chan})$
`fork`	$\forall(\alpha\beta : \emptyset).\ (\texttt{unit} \rightarrow^{\beta} \alpha) \rightarrow^{\emptyset} \texttt{unit}$

Figure 1: Type schemes for selected constants.

$t \equiv t'$, 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 depicted in Figure 3 and employs the notion of well-formedness: a constraint set is well-formed if all constraints are of form $t \subseteq \alpha$ or $b \subseteq \beta$; and 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 $\{\vec{\alpha}\vec{\beta}\}^{C_0\uparrow} = \{\vec{\alpha}\vec{\beta}\}$. Here[1]

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

where 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 $i = 1, 2$, and where we use $\leftarrow^*$ for the reflexive and transitive closure. Dually we have

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

Also we write $C \vdash C_0$ to mean that $C \vdash g_1 \subseteq g_2$ for all $g_1 \subseteq g_2$ in C_0 and 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 $C \vdash S_0\, C_0$.

[1]Following [3] 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

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

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

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

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

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

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

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

Ordering on types

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

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

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

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

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

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

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

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

Figure 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) $$\frac{C, A \vdash e \,:\, t \,\&\, b}{C, A \vdash e \,:\, t' \,\&\, b'}$$ if $C \vdash t \subseteq t'$ and $C \vdash b \subseteq b'$

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

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

Figure 3: The type inference system.

2.1 Properties of the Inference System

In this paper we shall use a number of technical results from [3]; to be self-contained we repeat their statements here.

Fact 2.2 Suppose $C \cup C_0 \vdash \gamma_1 \leftarrow \gamma_2$ with $\gamma_1 \notin FV(C)$. Then $C_0 \vdash \gamma_1 \leftarrow \gamma_2$.

Lemma 2.3 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 exist t_1 such that $t = t_1$ `chan` and such that $C \vdash t_1 \subseteq t'_1$ and $C \vdash t'_1 \subseteq t_1$.
- If $t' = t'_1$ `list` there exist 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`).

Lemma 2.4 Suppose that C is well-formed:

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

Lemma 2.5 *Substitution Lemma*

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).

Lemma 2.6 *Entailment Lemma*

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).

Fact 2.7 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).

Fact 2.8 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).

Recall from [3] that an inference tree is contraint-saturated whenever all occurrences of the rules (app), (let), and (if) have the same constraints in their premises. Next recall that a strongly normalised inference tree is a constraint-saturated inference tree whose structure essentially is that of the underlying expression: the rule (ins) is only allowed immediately after a (con) or (id), the rule (gen) is only allowed immediately before a `let` (and only in the left branch), and the rule (sub) is never allowed after a (gen) or (sub) and is required after all other rules; we refer to [3] for the precise definition.

Fact 2.9 *Enforcing Constraint-Saturation*

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).

Lemma 2.10 *Enforcing Strong Normalisation*

If A is well-formed and solvable from C then an inference tree $C, A \vdash e : \sigma \,\&\, b$ can be transformed into one $C, A \vdash_s e : \sigma \,\&\, b$ 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 by means of an "evaluation function" δ:

Definition 2.11 The function δ is a partial mapping from expressions into expressions: if $\delta(e)$ is defined then e will have the form $c\,e_1$ with c a sequential base function (but we do not claim that it is defined on all such arguments). It is defined by the following (incomplete) table:

c	e	$\delta(c\,e)$
`fst`	`pair` $e_1\,e_2$	e_1
`snd`	`pair` $e_1\,e_2$	e_2
`hd`	`cons` $e_1\,e_2$	e_1
`tl`	`cons` $e_1\,e_2$	e_2
`isnil`	`nil`	`true`
`isnil`	`cons` $e_1\,e_2$	`false`
`+`	`pair` $n_1\,n_2$	n where $n = n_1 + n_2$
$\vdots$	$\vdots$	

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

Definition 2.12 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 `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 employ the notion of *evaluation context*:

Definition 2.13 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'. The following (rather obvious) fact is proved in Appendix A:

Fact 2.14 $(E_1[E_2])[e] = E_1[E_2[e]]$.

Now we are ready for:

Definition 2.15 *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:

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

Fact 2.16 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 2.17 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 an easy induction on w. □

We shall say that an essentially closed expression e is *stuck* if it is not weakly evaluated and yet $e \not\rightarrow$. We shall say that a stuck expression e is *top-level stuck* if it cannot be written on the form $e = E[e']$ with $E \neq [\,]$ and with e' stuck. It is easy to see (using Fact 2.14) that for any stuck expression e there exists E and top-level stuck e' such that $e = E[e']$.

Fact 2.18 Suppose that e is essentially closed and top-level stuck; 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. If e is a constant, a channel identifier or an abstraction then e is weakly evaluated and hence not stuck. If e is of form $\texttt{rec}\ f\ x \Rightarrow e$, then $e \rightharpoonup \cdots$ and hence e is not stuck.

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 stuck (as e is top-level stuck). 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 stuck (as e is top-level stuck). 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 2.19 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, 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 2.20 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 the fact that δ is 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 2.21 *Concurrent Evaluation*

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

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

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

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

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

2.4 Manipulation of 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 2.22 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 clearly 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 2.23 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 2.24 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'.

Some of the subsequent proofs will be by induction in the depth of a judgement in an inference tree; for this purpose the following result is convenient:

Fact 2.25 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
> $E = E_2[E_1]$.

Proof We can clearly use jdg'' as in Definition 2.22. □

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 2.26 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_0 is such that $C, A \vdash e_0 : \sigma \,\&\, b$ then also $C', A' \vdash E[e_0] : \sigma' \,\&\, b'$.

Since the hole in an evaluation context is not inside the scope of any bound identifier we have:

Fact 2.27 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 also C is well-formed.

The (concurrent) transition which poses the greatest danger to semantic soundness is channel allocation, due to the need for an environment update (cf. the relationship to side effects in Standard ML). In order to construct an inference tree with the new environment we must demand that the type of the new channel is "present" in the behaviour:

Lemma 2.28 Suppose the judgement $C, A \vdash e : \sigma \,\&\, b$ occurs at E in the inference tree of $C', A \vdash E[e] : \sigma' \,\&\, b'$ where C' (and hence also C) is well-formed.

Let ch be a channel identifier not in $E[e]$, and let t be a type and e_0 an expression such that

> $C \vdash \{t \text{ CHAN}\} \subseteq b$ and $C, A[ch : t \texttt{ chan}] \vdash e_0 : \sigma \,\&\, b$.

Then it also holds that

> $C' \vdash \{t \text{ CHAN}\} \subseteq b'$ and $C', A[ch : t \texttt{ chan}] \vdash E[e_0] : \sigma' \,\&\, b'$.

Proof See Appendix A. □

Fact 2.27 told us something about the relationship between the root of an inference tree and the interior nodes of the tree. It proves useful to know some more:

Lemma 2.29 Suppose the judgement $C, A \vdash e : \sigma \,\&\, b$ occurs at E in the constraint-saturated inference tree of $C', A \vdash_c e' : \sigma' \,\&\, b'$ where C' (and hence also C) is well-formed.

Then $C' \subseteq C$, and there exists S with $Dom(S) \cap FV(A) = \emptyset$ such that $C' \vdash S\,C$.
Proof See Appendix A. □

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 2.30 Suppose that C is well-formed and that

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

and that TypeOf(c) is of form

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

where we demand 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

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

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 TypeOf(c) $= t'_1 \rightarrow^{b'_1} \;\; \cdots t'_m \rightarrow^{b'_m} \;\; t''$ 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 2.31 Suppose that $C, A[x : \sigma'] \vdash e : \sigma \,\&\, b$ and that $C, A \vdash e' : \sigma' \,\&\, \emptyset$; then $C, A \vdash e[e'/x] : \sigma \,\&\, b$.

Lemma 2.32 Suppose that $C, A \vdash_s \; w : \sigma \,\&\, b$ with C well-formed;
then $C, A \vdash w : \sigma \,\&\, \emptyset$ and with the same shape.

3 Semantic Soundness

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

Definition 3.1 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 going to be empty, and we shall see that the concurrent soundness result 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 3.2 Let C be well-formed and let A be a channel environment and suppose

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

If $e \rightharpoonup e'$ then also

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

Proof Due to Lemma 2.10 (which can be applied since A is trivially well-formed and solvable from C) we can assume that we in fact have $C, A \vdash_s e : \sigma \,\&\, b$. It will clearly suffice to show the result when σ is a type. Moreover, it will be enough if we can show the result in the case where the last application of the rule (sub) is a trivial one. We perform case analysis on the transition $\rightharpoonup$:

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

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

and using Lemma 2.32 we have

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

which by Lemma 2.31 can be combined with the second premise of the inference to yield

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

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

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

and using Lemma 2.32 we have

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

so by applying (rec) we get the judgement

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

which by Lemma 2.31 can be combined with the premise of the inference to yield the desired

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

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

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

and the claim is immediate.

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

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

where $C \vdash t_2' \rightarrow^{b_0'} \ t' \subseteq t_2 \rightarrow^{b_0} \ t$. Since C is well-formed we can apply Lemma 2.3 to deduce that

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

By Lemma 2.32 followed by an application of (sub) we get

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

which by Lemma 2.31 can be combined with the upmost leftmost premise of the inference to yield

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

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 the table defining δ; below we shall list some typical cases only. In all cases we make use of Lemma 2.30 and Lemma 2.3 which can be applied since C is well-formed.

$e = \mathtt{fst}\,(\mathtt{pair}\,e_1\,e_2)$ **and** $\delta(e) = e_1$**:** Then the situation is that

$C, A \vdash_s \mathtt{fst}\,(\mathtt{pair}\,e_1\,e_2) : t \,\&\, b$

so since TypeOf($\mathtt{fst}$) $= \forall(\alpha_1\alpha_2 : \emptyset).\ \alpha_1 \times \alpha_2 \to^{\emptyset} \alpha_1$ Lemma 2.30 tells us that there exists t_0, b_0 and S_0 such that

$C, A \vdash_s \mathtt{pair}\,e_1\,e_2 : t_0 \,\&\, b_0$ 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($\mathtt{pair}$) $= \forall(\alpha_1\alpha_2 : \emptyset).\ \alpha_1 \to^{\emptyset} \alpha_2 \to^{\emptyset} \alpha_1 \times \alpha_2$ Lemma 2.30 tells us that there exists t_1, b_1, t_2, b_2 and S such that

$C, A \vdash_s e_1 : t_1 \,\&\, b_1$ and $C, A \vdash_s e_2 : t_2 \,\&\, b_2$;

$C \vdash t_1 \subseteq S\,\alpha_1$ and $C \vdash t_2 \subseteq S\,\alpha_2$ and $C \vdash b_1 \subseteq b_0$ and $C \vdash b_2 \subseteq b_0$;

$C \vdash S\,(\alpha_1 \times \alpha_2) \subseteq t_0$.

Since $C \vdash t_1 \times t_2 \subseteq S\,\alpha_1 \times S\,\alpha_2 \subseteq t_0 \subseteq S_0\,\alpha_1 \times S_0\,\alpha_2$ we by Lemma 2.3 deduce that

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

and since $C \vdash b_1 \subseteq b_0 \subseteq b$ we from $C, A \vdash_s e_1 : t_1 \,\&\, b_1$ get the desired judgement

$C, A \vdash e_1 : t \,\&\, b$.

$e = +(\mathtt{pair}\, n_1\, n_2)$ **and** $\delta(e) = n$ **where** $n = n_1 + n_2$**:** Then the situation is that

$$C, A \vdash_s +(\mathtt{pair}\, n_1\, n_2) : t \,\&\, b$$

so since we have TypeOf$(+) = \mathtt{int} \times \mathtt{int} \rightarrow^{\emptyset} \mathtt{int}$ we can infer by Lemma 2.30 that

$$C \vdash \mathtt{int} \subseteq t.$$

But as $C \vdash \emptyset \subseteq b$ this is sufficient to show the desired judgement

$$C, A \vdash n : t \,\&\, b.$$

This completes the proof. □

Theorem 3.3 *Sequential soundness*

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

$$C, A \vdash e_1 : \sigma \,\&\, b.$$

If $e_1 \rightarrow e_2$ then also

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

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 2.24 there exists C', A', σ' and b' such that $C', A' \vdash e_1' : \sigma' \,\&\, b'$ occurs at E in the inference tree of $C, A \vdash E[e_1'] : \sigma \,\&\, b$. By Fact 2.27 we infer that $A' = A$ and that C' is well-formed. This enables us to use Lemma 3.2 from which we get

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

and by Fact 2.26 we get the desired judgement

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

This completes the proof. □

Remark. The purpose of types is to detect certain kinds of errors at analysis time rather than at execution time. To this end one usually wants a result that guarantees that "error configurations are not typeable"; here we presuppose some well-formed constraint set and some channel environment A. By Proposition 2.19 and the discussion after it, 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 2.3 and Lemma 2.30.

$ch\,w$ with ch a channel identifier: here we employ that $A(ch)$ is of form t `chan`.

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

> w can be assigned the type `bool`.
>
> Thus w cannot be a channel identifier (as A is a channel environment); w cannot be a function abstraction; and an examination of the function TypeOf 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 where $\delta(c\,w)$ is undefined: consider e.g. the expression `fst` w. For this to be typeable there must exist t_1 and t_2 such that

> w can be assigned the type $t_1 \times t_2$.
>
> Thus w cannot be a channel identifier (as A is a channel environment); w cannot be a function abstraction; and an examination of the function TypeOf 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`).

3.2 Concurrent Soundness

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

Lemma 3.4 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 $E[$`channel ()`$]$; then there exists t_0 such that

$$C \vdash \{t_0 \text{ CHAN}\} \subseteq b \text{ and}$$
$$C, A[ch : t_0 \texttt{ chan}] \vdash E[ch] : \sigma \,\&\, b.$$

Proof The strongly normalized inference tree contains a judgement of form

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

where C' is well-formed (Fact 2.27). Since

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

it follows from Lemma 2.30 that there exists 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 $t_0 = S\,\alpha$, then we have

$$C' \vdash \{t_0\ \text{CHAN}\} \subseteq b' \text{ and } C', A[ch : t_0\ \texttt{chan}] \vdash ch\ :\ t' \,\&\, b'$$

so by Lemma 2.28 we arrive at the desired relations

$$C \vdash \{t_0\ \text{CHAN}\} \subseteq b \text{ and } C, A[ch : t_0\ \texttt{chan}] \vdash E[ch]\ :\ \sigma \,\&\, b.$$

This completes the proof. □

Lemma 3.5 Let C be well-formed and suppose that

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

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

(a) $C, A \vdash E[\texttt{()}]\ :\ \sigma \,\&\, b$;

(b) $C, A \vdash e\,\texttt{()}\ :\ t'' \,\&\, b''$.

Proof The strongly normalized inference tree contains a judgement of form

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

and by Lemma 2.29 we infer that C' is well-formed and that there exists S' with $Dom(S') \cap FV(A) = \emptyset$ such that $C \vdash S'\,C'$.

Since $\text{TypeOf}(\texttt{fork}) = \forall(\alpha\beta : \emptyset).\ (\texttt{unit} \to^{\beta}\ \alpha) \to^{\emptyset}\ \texttt{unit}$ Lemma 2.30 tells us that there exists t_1, b_1 and S such that

$$C' \vdash \texttt{unit} \subseteq t' \tag{1}$$

$$C', A \vdash_s e : t_1 \,\&\, b_1 \text{ and } C' \vdash t_1 \subseteq \texttt{unit} \to^{S\beta} S\,\alpha \tag{2}$$

Here (1) (together with $C' \vdash \emptyset \subseteq b'$) tells us that

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

which by Fact 2.26 yields claim (a).

For claim (b) we use Lemma 2.3 on (2) to find t_1', b_1', t_1'' such that

$$C', A \vdash_s e : t_1' \to^{b_1'} t_1'' \,\&\, b_1 \text{ and } C' \vdash \texttt{unit} \subseteq t_1'.$$

This shows that

$$C', A \vdash e\,\texttt{()} : t_1'' \,\&\, b_1'' \text{ for some } b_1''$$

and by Lemma 2.5 and Lemma 2.6 (since $C \vdash S'\,C'$) this yields the desired judgement

$$C, A \vdash e\,\texttt{()} : S'\,t_1'' \,\&\, S'\,b_1''.$$

This completes the proof. □

Lemma 3.6 Let C be well-formed and let A be a channel environment and suppose that

$$C, A \vdash_s E_1[\texttt{sync}\,(\texttt{send}\,(\texttt{pair}\,ch\,w))] : \sigma_1 \,\&\, b_1 \tag{3}$$

and suppose that

$$C, A \vdash_s E_2[\texttt{sync}\,(\texttt{receive}\,ch)] : \sigma_2 \,\&\, b_2. \tag{4}$$

Let $A(ch) = t$ `chan`, then

(a) $C, A \vdash E_1[w] : \sigma_1 \,\&\, b_1$;

(b) $C, A \vdash w : t \,\&\, \emptyset$;

(c) $C, A \vdash E_2[w] : \sigma_2 \,\&\, b_2$.

Proof The tree (3) will contain a judgement of form

$$C_1, A \vdash_s \texttt{sync}\,(\texttt{send}\,(\texttt{pair}\,ch\,w)) \,:\, t_1 \,\&\, b_1' \tag{5}$$

with C_1 well-formed. Since TypeOf($\texttt{sync}$) $= \forall(\alpha\beta : \emptyset).\ (\alpha\ \texttt{com}\ \beta) \rightarrow^{\beta}\ \alpha$ Lemma 2.30 tells us that there exists t_3, b_3 and S_3 such that

$C_1, A \vdash_s \texttt{send}\,(\texttt{pair}\,ch\,w) \,:\, t_3 \,\&\, b_3$;
$C_1 \vdash S_3\,\alpha \subseteq t_1$;
$C_1 \vdash t_3 \subseteq (S_3\,\alpha)\ \texttt{com}\ (S_3\,\beta)$;
$C_1 \vdash b_3 \subseteq b_1'$.

Since TypeOf($\texttt{send}$) $= \forall(\alpha : \emptyset).\ \alpha\ \texttt{chan} \times \alpha \rightarrow^{\emptyset}\ \alpha\ \texttt{com}\ \emptyset$ Lemma 2.30 tells us that there exists t_4, b_4 and S_4 such that

$C_1, A \vdash_s \texttt{pair}\,ch\,w \,:\, t_4 \,\&\, b_4$;
$C_1 \vdash (S_4\,\alpha)\ \texttt{com}\ \emptyset \subseteq t_3$;
$C_1 \vdash t_4 \subseteq (S_4\,\alpha)\ \texttt{chan} \times (S_4\,\alpha)$;
$C_1 \vdash b_4 \subseteq b_3$.

Since TypeOf($\texttt{pair}$) $= \forall(\alpha_1\alpha_2 : \emptyset).\ \alpha_1 \rightarrow^{\emptyset}\ \alpha_2 \rightarrow^{\emptyset}\ \alpha_1 \times \alpha_2$ Lemma 2.30 tells us that there exists t_5, b_5, t_6, b_6 and S_5 such that

$C_1, A \vdash_s ch \,:\, t_5 \,\&\, b_5$; (6)
$C_1, A \vdash_s w \,:\, t_6 \,\&\, b_6$; (7)
$C_1 \vdash S_5\,\alpha_1 \times S_5\,\alpha_2 \subseteq t_4$;
$C_1 \vdash t_5 \subseteq S_5\,\alpha_1$ and $C_1 \vdash t_6 \subseteq S_5\,\alpha_2$;
$C \vdash b_6 \subseteq b_4$.

Since $A(ch) = t\ \texttt{chan}$ we infer from (6) that

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

We now apply Lemma 2.3 repeatedly: from

$C_1 \vdash (S_4\,\alpha)\ \texttt{com}\ \emptyset \subseteq t_3 \subseteq (S_3\,\alpha)\ \texttt{com}\ (S_3\,\beta)$ and
$C_1 \vdash t_5 \times t_6 \subseteq S_5\,\alpha_1 \times S_5\,\alpha_2 \subseteq t_4 \subseteq (S_4\,\alpha)\ \texttt{chan} \times (S_4\,\alpha)$

we deduce that

$C_1 \vdash S_4\,\alpha \subseteq S_3\,\alpha \subseteq t_1$ and
$C_1 \vdash t\ \texttt{chan} \subseteq t_5 \subseteq (S_4\,\alpha)\ \texttt{chan}$ and
$C_1 \vdash t_6 \subseteq S_4\,\alpha$.

By applying Lemma 2.3 once more[2], exploiting the *contravariance* of $\cdots$ `chan` (cf. the remarks concerning Figure 2), we end up with the following relations:

$$C_1 \vdash t_6 \subseteq t_1 \text{ and } C_1 \vdash t_6 \subseteq t.$$

As $C_1 \vdash b_6 \subseteq b_1'$ we get from (7) that

$$C_1, A \vdash w \,:\, t_1 \,\&\, b_1',$$

which by Fact 2.26 yields claim (a); next using Lemma 2.32 on (7) we also get

$$C_1, A \vdash w \,:\, t \,\&\, \emptyset.$$

By Lemma 2.29 there exists S_1 with $Dom(S_1) \cap FV(A) = \emptyset$ such that $C \vdash S_1\, C_1$, so by applying Lemma 2.5 and Lemma 2.6 we arrive at

$$C, A \vdash w \,:\, S_1\, t \,\&\, \emptyset$$

which yields the claim (b) since $FV(t) \subseteq FV(A)$ and hence $S_1\, t = t$.

Our remaining task is to show claim (c), where we first notice that the tree (4) will contain a judgement of form

$$C_2, A \vdash_s \texttt{sync}\,(\texttt{receive}\, ch) \,:\, t_2 \,\&\, b_2' \tag{8}$$

with C_2 well-formed. Since TypeOf(`sync`) $= \forall(\alpha\beta : \emptyset).\ (\alpha\ \texttt{com}\ \beta) \rightarrow^{\beta}\ \alpha$ Lemma 2.30 tells us that there exists t_7, b_7 and S_7 such that

$$\begin{array}{l} C_2, A \vdash_s \texttt{receive}\, ch \,:\, t_7 \,\&\, b_7; \\ C_2 \vdash S_7\, \alpha \subseteq t_2; \\ C_2 \vdash t_7 \subseteq (S_7\, \alpha)\ \texttt{com}\ (S_7\, \beta). \end{array}$$

Since TypeOf(`receive`) $= \forall(\alpha : \emptyset).\ (\alpha\ \texttt{chan}) \rightarrow^{\emptyset}\ (\alpha\ \texttt{com}\ \emptyset)$ Lemma 2.30 tells us that there exists t_8, b_8 and S_8 such that

$$\begin{array}{l} C_2, A \vdash_s ch \,:\, t_8 \,\&\, b_8; \\ C_2 \vdash (S_8\, \alpha)\ \texttt{com}\ \emptyset \subseteq t_7; \\ C_2 \vdash t_8 \subseteq (S_8\, \alpha)\ \texttt{chan}. \end{array} \tag{9}$$

Since $A(ch) = t$ `chan` we infer from (9) that

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

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

We now apply Lemma 2.3 repeatedly: from

$$C_2 \vdash (S_8\,\alpha) \texttt{ com } \emptyset \subseteq t_7 \subseteq (S_7\,\alpha) \texttt{ com } (S_7\,\beta) \text{ and}$$
$$C_2 \vdash t \texttt{ chan} \subseteq t_8 \subseteq (S_8\,\alpha) \texttt{ chan}$$

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

$$C_2 \vdash t \subseteq S_8\,\alpha \subseteq S_7\,\alpha \subseteq t_2.$$

Since Lemma 2.29 ensures that $C \subseteq C_2$ we can deduce from claim (b) that

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

so by applying (sub) we arrive at

$$C_2, A \vdash w : t_2 \,\&\, b_2'$$

which by Fact 2.26 yields claim (c). □

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 $C, A \vdash PP(p) : PT(p) \,\&\, PB(p)$ for all $p \in Dom(PP)$.

Theorem 3.7 *Semantic (concurrent) soundness*

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

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

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

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

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

Furthermore we have the following property:

- if $a = p$ `chan` ch then there exists t_0 such that $C \vdash \{t_0 \text{ CHAN}\} \subseteq PB(p)$ and such that $A'(ch) = t_0$ `chan`.

Proof Notice that by Lemma 2.10 we can assume that the inference trees in $C, A \vdash PP : PT \,\&\, PB$ are strongly normalised. We perform case analysis on the action label a:

$a = \texttt{seq}$: It follows from Theorem 3.3 that we can use $PT' = PT$, $PB' = PB$ and $A' = A$.

$a = p$ `chan` ch: It follows from Lemma 3.4 that there exists t_0 such that the claim follows with $PT' = PT$, $PB' = PB$ and $A' = A[ch : t_0\ \texttt{chan}]$. (For p' in the domain of PP with $p' \neq p$ we must show that $C, A \vdash PP(p') : PT(p') \,\&\, PB(p')$ implies $C, A' \vdash PP(p') : PT(p') \,\&\, PB(p')$, but this follows from Fact 2.8.)

$a = p$ `fork` p': It follows from Lemma 3.5 that there exists t'', b'' such that we can use $PT' = PT[p' : t'']$, $PB' = PB[p' : b'']$ and $A' = A$.

$a = \texttt{comm}$: It follows from Lemma 3.6 that we can use $PT' = PT$, $PB' = PB$ and $A' = A$. □

Remark. Theorem 3.7 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 3.6 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 3.6 does not directly relate:

- the type t_6 of the value w actually communicated (see line 7),
- 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 5), and
- the type t_2 that the receiver thinks was communicated (see line 8).

However, by inspecting the proof of Lemma 3.6 one may note that the following relations are established:

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

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 [3]. Although the development was performed for a fragment of Concurrent ML we believe it equally possible for Standard ML with references.

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.

References

[1] M.Felleisen, D.P.Friedman: Control Operators, the SECD-Machine, and the λ-calculus. *Formal Descriptions of Programming Concepts III*, North-Holland, 1986.

[2] H.R. Nielson and F. Nielson. Higher-order Concurrent Programs with Finite Communication Topology. In *Proc. POPL'94*, pages 84–97. ACM Press, 1994.

[3] H.R.Nielson, F.Nielson, T.Amtoft: Polymorphic Subtypes for Effect Analysis: the Integration, 1996.

[4] G.D.Plotkin: A Structural Approach to Operational Semantics, Report DAIMI FN-19, Aarhus University, Denmark, 1981.

[5] J. H. Reppy. Concurrent ML: Design, Application and Semantics. In *Proc. Functional Programming, Concurrency, Simulation and Automated Reasoning*, pages 165–198. SLNCS 693, 1993.

A Details of Proofs

The sequential semantics

Fact 2.14 $(E_1[E_2])[e] = E_1[E_2[e]]$.

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

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

This completes the proof. □

Manipulation of proof trees

Fact 2.24 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'.

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. 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 2.28 Suppose the judgement $jdg = (C, A \vdash e : \sigma \,\&\, b)$ occurs at E with depth n in the inference tree of $jdg' = (C', A \vdash E[e] : \sigma' \,\&\, b')$ where C' (and hence also C) is well-formed.

Let ch be a channel identifier not in $E[e]$, and let t be a type and e_0 an expression such that

$$C \vdash \{t \text{ CHAN}\} \subseteq b \text{ and } C, A[ch : t \texttt{ chan}] \vdash e_0 : \sigma \,\&\, b.$$

Then it also holds that

$$C' \vdash \{t \text{ CHAN}\} \subseteq b' \text{ and } C', A[ch : t \texttt{ chan}] \vdash E[e_0] \,:\, \sigma' \,\&\, b'.$$

Proof We perform induction in n: if $n = 0$ then $E = [\,]$, $C' = C$, $\sigma' = \sigma$, $b' = b$ and the claim is trivial.

If $n > 1$ then by Fact 2.25 there exists judgement $jdg'' = C'', A'' \vdash e'' \,:\, \sigma'' \,\&\, b''$ and evaluation contexts E_1 and E_2 such that $E = E_2[E_1]$ and 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'.

So if $C \vdash \{t \text{ CHAN}\} \subseteq b$ and $C, A[ch : t \texttt{ chan}] \vdash e_0 \,:\, \sigma \,\&\, b$ we can apply the induction hypothesis (with jdg and jdg'') to infer that $C'' \vdash \{t \text{ CHAN}\} \subseteq b''$ and that $C'', A[ch : t \texttt{ chan}] \vdash E_1[e_0] \,:\, \sigma'' \,\&\, b''$; and by applying the induction hypothesis once more (with jdg'' and jdg') we can infer $C' \vdash \{t \text{ CHAN}\} \subseteq b'$ and $C', A[ch : t \texttt{ chan}] \vdash E_2[E_1[e_0]] \,:\, \sigma' \,\&\, b'$ which is as desired (due to Fact 2.14).

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:

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

and our assumptions are

$$C_1 \vdash \{t \text{ CHAN}\} \subseteq b_1 \text{ and } C_1, A[ch : t \texttt{ chan}] \vdash e_0 \,:\, t_2 \to^b t_1 \,\&\, b_1$$

and we must show that

> $C_1 \cup C_2 \vdash \{t \text{ CHAN}\} \subseteq b_1 \cup b_2 \cup b$ and
> $(C_1 \cup C_2), A[ch : t \texttt{ chan}] \vdash e_0\, e_2 \,:\, t_1 \,\&\, (b_1 \cup b_2 \cup b)$.

The former is a trivial consequence of the assumptions, and the latter will follow provided we can show that $C_2, A[ch : t \texttt{ chan}] \vdash e_2 \,:\, t_2 \,\&\, b_2$. But this follows from Fact 2.8 since ch does not occur in e_2.

$\underline{E = w\, E_2}$**:** Similar to the case above (now exploiting that for all C it holds that $C \vdash b_2 \subseteq b_1 \cup b_2 \cup b$).

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

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

and our assumptions are

$$C_1 \vdash \{t\ \textsc{chan}\} \subseteq b_1 \text{ and } C_1, A[ch : t\ \texttt{chan}] \vdash e_0 : ts_1 \,\&\, b_1$$

and we must show that

$$C_1 \cup C_2 \vdash \{t\ \textsc{chan}\} \subseteq b_1 \cup b_2 \text{ and}$$
$$(C_1 \cup C_2), A[ch : t\ \texttt{chan}] \vdash \texttt{let}\ x = e_0\ \texttt{in}\ e_2 : t_2 \,\&\, (b_1 \cup b_2).$$

The former is a trivial consequence of the assumptions, and the latter will follow provided we can show that $C_2, A[ch : t\ \texttt{chan}][x : ts_1] \vdash e_2 : t_2 \,\&\, b_2$. But this follows from Fact 2.8 and Fact 2.7 since $ch \neq x$ and ch does 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:

$$\frac{jdg = C_0, A \vdash e_0 : \texttt{bool} \,\&\, b_0 \qquad C_1, A \vdash e_1 : t_1 \,\&\, b_1 \qquad C_2, A \vdash e_2 : t_1 \,\&\, b_2}{jdg' = (C_0 \cup C_1 \cup C_2), A \vdash \texttt{if}\ e_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2 : t_1 \,\&\, (b_0 \cup b_1 \cup b_2)}$$

and our assumptions are

$$C_0 \vdash \{t\ \textsc{chan}\} \subseteq b_0 \text{ and } C_0, A[ch : t\ \texttt{chan}] \vdash e_0 : \texttt{bool} \,\&\, b_0$$

and we must show that

$$C_0 \cup C_1 \cup C_2 \vdash \{t\ \textsc{chan}\} \subseteq b_0 \cup b_1 \cup b_2 \text{ and}$$
$$(C_0 \cup C_1 \cup C_2), A[ch : t\ \texttt{chan}] \vdash \texttt{if}\ e_0\ \texttt{then}\ e_1\ \texttt{else}\ e_2 : t_1 \,\&\, (b_0 \cup b_1 \cup b_2).$$

The former is a trivial consequence of the assumptions, and the latter will follow if we can show $C_1, A[ch : t\ \texttt{chan}] \vdash e_1 : t_1 \,\&\, b_1$ and $C_2, A[ch : t\ \texttt{chan}] \vdash e_2 : t_1 \,\&\, b_2$. But this follows from Fact 2.8 since ch does not occur in e_1 or e_2.

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

(sub) has been applied: the situation is

$$\frac{jdg = C, A \vdash e \,:\, t_0 \,\&\, b}{jdg' = C, A \vdash e \,:\, t' \,\&\, b'}$$

where $C \vdash t_0 \subseteq t'$ and $C \vdash b \subseteq b'$. Our assumptions are

$$C \vdash \{t \text{ CHAN}\} \subseteq b \text{ and } C, A[ch : t \texttt{ chan}] \vdash e_0 \,:\, t_0 \,\&\, b$$

and we must show that

$$C \vdash \{t \text{ CHAN}\} \subseteq b' \text{ and } C, A[ch : t \texttt{ chan}] \vdash e_0 \,:\, t' \,\&\, b'.$$

But this is trivial.

(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. Our assumptions are

$$C \vdash \{t \text{ CHAN}\} \subseteq b \text{ and } C, A[ch : t \texttt{ chan}] \vdash e_0 \,:\, \forall(\vec{\alpha}\vec{\beta} : C_0).\, t_0 \,\&\, b$$

and we must show that

$$C \vdash \{t \text{ CHAN}\} \subseteq b \text{ and } C, A[ch : t \texttt{ chan}] \vdash e_0 \,:\, S_0\, t_0 \,\&\, b.$$

But this is trivial.

(gen) has been applied: this is the really interesting case! 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 with $Dom(S) \subseteq \{\vec{\alpha}\vec{\beta}\}$ such that $C \vdash S\, C_0$. Our assumptions are

$$C \,\cup\, C_0 \vdash \{t \text{ CHAN}\} \subseteq b \tag{1}$$
$$C \,\cup\, C_0, A[ch : t \texttt{ chan}] \vdash e_0 \,:\, t_0 \,\&\, b \tag{2}$$

and we must show that

$$C \vdash \{t \text{ CHAN}\} \subseteq b \tag{3}$$
$$C, A[ch : t \texttt{ chan}] \vdash e_0 \; : \; \forall(\vec{\alpha}\vec{\beta} : C_0).\; t_0 \,\&\, b \tag{4}$$

It will suffice to prove

$$\{\vec{\alpha}\vec{\beta}\} \cap FV(t) = \emptyset \tag{5}$$

for then (1) and Lemma 2.5(a) give that $C \cup S\,C_0 \vdash t \text{ CHAN} \subseteq b$ which (by Lemma 2.6) implies (3); and we will be able to use (gen) to arrive at (4) from (2).

So we are left with the task of proving (5). Since $\forall(\vec{\alpha}\vec{\beta} : C_0).\; t_0$ is well-formed it holds that $\{\vec{\alpha}\vec{\beta}\}^{C_0\uparrow} = \{\vec{\alpha}\vec{\beta}\}$, that is

$$\text{if } C_0 \vdash \gamma \leftarrow \gamma' \text{ with } \gamma \in \{\vec{\alpha}\vec{\beta}\} \text{ then also } \gamma' \in \{\vec{\alpha}\vec{\beta}\}. \tag{6}$$

By Fact 2.2 we are able to infer (as $\{\vec{\alpha}\vec{\beta}\} \cap FV(C) = \emptyset$) that

$$\text{if } C \cup C_0 \vdash \gamma \leftarrow \gamma' \text{ with } \gamma \in \{\vec{\alpha}\vec{\beta}\} \text{ then } C_0 \vdash \gamma \leftarrow \gamma'. \tag{7}$$

By combining (6) and (7) we infer that

$$\text{if } C \cup C_0 \vdash \gamma \leftarrow \gamma' \text{ with } \gamma \in \{\vec{\alpha}\vec{\beta}\} \text{ then also } \gamma' \in \{\vec{\alpha}\vec{\beta}\}$$

which amounts to $\{\vec{\alpha}\vec{\beta}\}^{(C \cup C_0)\uparrow} = \{\vec{\alpha}\vec{\beta}\}$. Since $\{\vec{\alpha}\vec{\beta}\} \cap FV(b) = \emptyset$ we have $\{\vec{\alpha}\vec{\beta}\}^{(C \cup C_0)\uparrow} \cap FV(b) = \emptyset$, and hence we do not have $C \cup C_0 \vdash \gamma \leftarrow^* \gamma'$ for any $\gamma \in \{\vec{\alpha}\vec{\beta}\}$ and $\gamma' \in FV(b)$. But this is just another way of saying that

$$\{\vec{\alpha}\vec{\beta}\} \cap FV(b)^{(C \cup C_0)\downarrow} = \emptyset. \tag{8}$$

From (1) and from Lemma 2.4 (which can be applied since we know that $C \cup C_0$ is well-formed) we infer that

$$FV(t)^{(C \cup C_0)\downarrow} \subseteq FV(b)^{(C \cup C_0)\downarrow}. \tag{9}$$

Combining (8) and (9) we get $\{\vec{\alpha}\vec{\beta}\} \cap FV(t)^{(C \cup C_0)\downarrow} = \emptyset$ which trivially implies (5). This completes the proof. □

Lemma 2.29 Suppose the judgement $jdg = C, A \vdash e \; : \; \sigma \,\&\, b$ occurs at E with depth n in the constraint-saturated inference tree of $jdg' = C', A \vdash_c e' \; : \; \sigma' \,\&\, b'$ where C' (and hence also C) is well-formed.

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

Proof We perform induction in n: if $n = 0$ then $C' = C$ and we can use $S = \mathrm{Id}$.

If $n > 1$ then by Fact 2.25 there exists judgement $jdg'' = C'', A'' \vdash e'' : \sigma'' \,\&\, b''$ 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'.

We can thus apply the induction hypothesis twice to infer that $C' \subseteq C'' \subseteq C$ and that there exists S_1,S_2 with $Dom(S_1) \cap FV(A) = \emptyset = Dom(S_2) \cap FV(A)$ such that $C' \vdash S_2\,C''$ and $C'' \vdash S_1\,C$. But then we by Lemma 2.5 and 2.6 arrive at $C' \vdash S_2\,S_1\,C$, where clearly $Dom(S_2\,S_1) \cap FV(A) = \emptyset$.

So we are left with the case $n = 1$. We perform case analysis on the inference rule applied. The only interesting case is (gen), for otherwise we have $C' = C$ due to our assumption about the inference tree being constraint saturated and hence we can use $S = \mathrm{Id}$. The situation thus 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 $\{\vec{\alpha}\vec{\beta}\} \cap FV(C, A, b) = \emptyset$ and where there exists S with $Dom(S) \subseteq \{\vec{\alpha}\vec{\beta}\}$ such that $C \vdash S\,C_0$. Our task can be accomplished by showing that $C \subseteq C \cup C_0$ and that $Dom(S) \cap FV(A) = \emptyset$ and that $C \vdash S\,C_0$ and that $C \vdash S\,C$. But all this follows directly. □

Lemma 2.30 Suppose that C is well-formed and that

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

and that TypeOf(c) is of form

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

where we demand 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

$$\mathrm{TypeOf}(c) = \forall(\vec{\alpha}\vec{\beta} : C_0).\ t'_1 \rightarrow^{b'_1} \cdots t'_n \rightarrow^{b'_n} t' \tag{10}$$

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$(c) = t'_1 \rightarrow^{b'_1} \cdots t'_m \rightarrow^{b'_m} t''$ in which case $\{\vec{\alpha}\vec{\beta}\} = \emptyset$ and $C_0 = \emptyset$ (so we have $S = \mathrm{Id}$).

Proof We perform induction in n. If $n = 0$ we can trivially always assume (10), 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 an application 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

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

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 $\forall(\vec{\alpha}\vec{\beta} : C_0).\ t'_1 \rightarrow^{b'_1} \cdots t'_{n-1} \rightarrow^{b'_{n-1}} t'''$

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

$$\begin{array}{l} Dom(S) \subseteq \{\vec{\alpha}\vec{\beta}\} \text{ and } C \vdash S\,C_0; \\ C \vdash S\,t''' \subseteq t_n \rightarrow^{b''} t^-; \\ \text{for all } i \in \{1 \cdots n-1\}\text{: } C, A \vdash_s e_i : t_i \,\&\, b_i \text{ and } C \vdash t_i \subseteq S\,t'_i \text{ and} \\ C \vdash b_i \subseteq b' \subseteq b \text{ and } C \vdash S\,b'_i \subseteq b' \subseteq b. \end{array} \qquad (11)$$

Since C is well-formed we can apply Lemma 2.3 on (11) to infer that $S\,t'''$ is a function type. If c is a constructor Fact 2.1 tells us that t''' cannot be a variable; hence in all cases we can write $t''' = t'_n \rightarrow^{b'_n} t'$ which amounts to (10). Lemma 2.3 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 2.31 Suppose that $C, A[x : \sigma'] \vdash e : \sigma \,\&\, b$ and that $C, A \vdash e' : \sigma' \,\&\, \emptyset$; then $C, A \vdash e[e'/x] : \sigma \,\&\, b$.

Proof Induction in the shape of the proof tree for $C, A[x : \sigma'] \vdash e : \sigma \,\&\, b$ which we by Fact 2.9 can assume to be constraint saturated. We perform case analysis on the last rule applied:

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

(id) 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 $\sigma = \sigma'$ and $b = \emptyset$. Since $e[e'/x] = e'$ the claim follows from the second part of the assumption.

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

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

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 2.7 and Fact 2.8 to get

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

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

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

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

(app) has been applied: Here the inference (which was assumed to be constraint saturated) takes the form

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

where we can apply the induction hypothesis twice and subsequently use (app) to construct an inference tree whose last inference is

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

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 2.7 and Fact 2.8 and we only spell the case (rec) out in detail. Here the inference takes the form

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

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 2.7 and Fact 2.8 to get

> $C, A[f : t][x : \sigma'] \vdash \texttt{fn}\ y \Rightarrow e\ :\ t \,\&\, b$ with the same shape as the premise and $C, A[f : t] \vdash e'\ :\ \sigma' \,\&\, \emptyset$.

We can thus apply the induction hypothesis to infer

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

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

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

By applying (rec) we get

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

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

$$\frac{C, A[x : \sigma'] \vdash e\ :\ t \,\&\, b}{C, A[x : \sigma'] \vdash e\ :\ t' \,\&\, b'} \qquad \text{with } C \vdash t \subseteq t' \text{ and } C \vdash b \subseteq b'$$

so we can apply the induction hypothesis and subsequently use (sub) to construct an inference tree whose last inference is

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

(ins) has been applied: Similar to the above case.

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

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

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 2.6 we have

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

so we can apply the induction hypothesis to get

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

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

Lemma 2.32 Suppose that $C, A \vdash_s w : \sigma \,\&\, b$ with C well-formed; then $C, A \vdash_s w : \sigma \,\&\, \emptyset$ and with the same shape.

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 2.30 combined with Fact 2.1 tells us that

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

and Lemma 2.30 further tells us that there exists $t_1 \cdots t_n$, $b_1 \cdots b_n$, and S with $Dom(S) \subseteq \{\vec{\alpha}\vec{\beta}\}$ such that

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

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

The induction hypothesis tells us that

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

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

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

This shows that we can construct the desired judgement $C, A \vdash_s c\, w_1 \cdots w_n : t \,\&\, \emptyset$ (where the applications of (sub) are justified by $C \vdash \emptyset \cup \emptyset \cup \emptyset \subseteq \emptyset$). □