Coinduction and Bisimilarity

University of Leicester, UK

Introduction

- Security often involves processes which can communicate.
- One wants to know if a particular communication is secure, perhaps in the sense that some data is kept private to certain individuals, and cannot be revealed to an environment.
- To do this, it can be useful to have a way of comparing processes, and describing when they are equivalent.
- These lectures describe a general theory of equivalence, illustrate applications of the theory within functional programming, and briefly survey some papers on security.

Overview Part I

- We review ordered sets. An order, or "comparison", can be used to generate equivalences.
- We discuss inductively, and coinductively defined sets. Such sets arise naturally when defining, and reasoning about, programs and processes.
- We define proof principles for such sets.
- We use the principle of coinduction to validate equivalences, and discuss when this is possible.

What's Next?

- We want to be able to show that program and process expressions are equivalent in some sense.
- To do this, we first try to order expressions in a sensible way.
- Thus we recall the notion of order and the (possibly derived) notion of equivalence relation.
- These can often be defined as fixed points.

Elementary Order Theory

- Consider $(\mathbb{N}, \leq)$ and its properties.

- A binary relation $\mathcal{R}$ on a set P is a preorder if it is reflexive and transitive; and
  - if also symmetric, then $\mathcal{R}$ is an equivalence relation;
  - if also $x \mathrel{\mathcal{R}} y \wedge y \mathrel{\mathcal{R}} x \Longrightarrow x = y$, anti-symmetry, then $\mathcal{R}$ is a partial order.

- A preordered/partially ordered set or preset/poset is a pair $(P, \mathcal{R})$ where P is a set and $\mathcal{R}$ is a preorder/partial order on P.

- If $S \subseteq P$ then we write $\bigwedge S$ for the greatest lower bound of S; dually we write $\bigvee S$ for the least upper bound of S, that is

$$l \leq \bigwedge S \iff (\forall x \in S)(l \leq x) \qquad \bigvee S \leq u \iff (\forall x \in S)(x \leq u)$$

- Key example: Powerset $(\mathcal{P}(X), \subseteq)$ of all subsets of X. In $\mathcal{P}(\mathbb{N})$, for example,

$$\bigvee \{\, \{n\} \mid n \leq 5 \,\} = \bigcup \{\, \{n\} \mid n \leq 5 \,\} = \{\, 1, \ldots, 5 \,\}$$

- P is called a complete lattice if joins of all subsets S exist or (equivalently) the meets of all subsets exist. $\mathcal{P}(X)$ is a complete lattice as all unions exist.

Fixed Points

- Endofunction $\Phi : P \to P$ between presets is monotone just in case it preserves the order: $x \leq y \Longrightarrow \Phi(x) \leq \Phi(y)$.

- If $P \stackrel{\text{def}}{=} \mathcal{P}(\{1,2,3\})$ and $\Phi(S) \stackrel{\text{def}}{=} S \cup \{2\}$, then

$$\Phi \text{ is monotone} \qquad \Phi(\{2\}) = \{2\} \qquad \{1\} \subseteq \Phi(\{1\})$$

- If $x \in P$ then we call x
  - a fixed point for Φ if $\Phi(x) = x$;
  - a pre-fixed point of Φ if $\Phi(x) \leq x$; and
  - a post-fixed point of Φ if $x \leq \Phi(x)$.

If P is a complete lattice (eg powerset), and $\Phi: P \to P$ is monotone:

- the least pre-fixed point exists:

$$\mu\Phi \stackrel{\text{def}}{=} \underbrace{\bigwedge\{\, x \in P \mid \Phi(x) \leq x \,\}}_{\text{NB! greatest lower bound}}$$

- the greatest post-fixed point exists:

$$\nu\Phi \stackrel{\text{def}}{=} \underbrace{\bigvee\{\, x \in P \mid x \leq \Phi(x) \,\}}_{\text{NB! least upper bound}}$$

Note: $\mu\Phi$ and $\nu\Phi$ are both fixed points. *Exercise*: use the definitions.

What's Next?

- Rules "connect" two pieces of data, a hypothesis and conclusion: eg $(\varnothing, 1), (z, z*2)$ over $\mathbb{Z}$.
- The smallest set of data such that if any hypothesis is a datum, then the conclusion is also a datum, is a pervasive notion in computing. Such sets are said to be inductively defined; eg $\mu = 1, 2, 4, 8, 16, \ldots$.
- The greatest set of data such that if any datum is the conclusion of a rule, then the hypothesis is also a datum, is also pervasive. Such sets are said to be coinductively defined; eg $\nu = 0, 1, 2, 4, 8, 16, \ldots$.

(Co)Inductively Defined Sets

- If $P \stackrel{\text{def}}{=} \mathcal{P}(\{1,2,3\})$ and $\Phi(S) \stackrel{\text{def}}{=} S \cup \{2\}$, then

$$\mu\Phi = \{2\} \qquad\qquad \nu\Phi = \{1,2,3\}$$

- Given $\Phi : \mathcal{P}(X) \to \mathcal{P}(X)$ monotone,
  - the $\boxed{\text{subset of } X \text{ inductively defined by } \Phi \text{ is } \mu\Phi}$;
  - the $\boxed{\text{subset of } X \text{ coinductively defined by } \Phi \text{ is } \nu\Phi}$.

Rule Notation

- A set of rules $\mathbf{R}$ on X is any subset

$$\mathbf{R} \subseteq \mathcal{P}(X) \times X$$

- We can write finitary rules like this
  - a base rule $R = (\varnothing, c)$

$$\frac{}{c}\, R$$

  - and an inductive rule $R = (H, c) = (\{h_1, \ldots, h_k\}, c)$

$$\frac{h_1 \quad h_2 \quad \ldots \quad h_k}{c}\, R$$

- The name of $\mathbf{R}$ is the function $\Phi_{\mathbf{R}} : \mathcal{P}(X) \to \mathcal{P}(X)$ given by setting

$$\Phi_{\mathbf{R}}(S) \stackrel{\text{def}}{=} \{\, x \in X \mid \exists \frac{S'}{x} \in \mathbf{R} \wedge S' \subseteq S \,\}$$

Informally: $x \in \Phi_{\mathbf{R}}(S)$ if x concludes a rule with hypotheses in S.

Exercise: Check monotone, and that any Φ arises as some $\Phi_{\mathbf{R}}$.

- Given X, and $\mathbf{R}$ on X,
  - the $\boxed{\text{subset of } X \text{ inductively defined by } \mathbf{R} \text{ is } \mu\Phi_{\mathbf{R}}}$;
  - the $\boxed{\text{subset of } X \text{ coinductively defined by } \mathbf{R} \text{ is } \nu\Phi_{\mathbf{R}}}$.

Examples of (Co)Inductively Defined Sets

Consider $\mathbf{R} \subseteq \mathcal{P}(\mathbb{Z}) \times \mathbb{Z}$ given by $\overline{\;0\;}$ and $\frac{z}{z+1}$. Then

$$\begin{aligned} \Phi_{\mathbf{R}}(S) &= \{\, n \in \mathbb{Z} \mid \frac{}{n=0} \vee (\exists z)(\frac{z}{n=z+1} \wedge \{z\} \subseteq S) \,\} \\ &= \{\, n \in \mathbb{Z} \mid n = 0 \vee (\exists z \in S)(n = z+1) \,\} \\ &= \{0\} \cup \{\, z+1 \mid z \in S \,\} \end{aligned}$$

Thus $\mu\Phi_{\mathbf{R}} = \mathbb{N}$ is least such that $\Phi_{\mathbf{R}}(S) \subseteq S$, and $\nu\Phi_{\mathbf{R}} = \mathbb{Z}$ is greatest such that $S \subseteq \Phi_{\mathbf{R}}(S)$, as if $m \in \mathbb{Z}$ then

$$m = (m-1)+1 \in \{\, z+1 \mid z \in \mathbb{Z} \,\} \subseteq \Phi_{\mathbf{R}}(\mathbb{Z})$$

Fix $\mathcal{R} \subseteq A \times A$. Consider $\mathbf{R} \subseteq \mathcal{P}(A) \times A$ given by

$$\frac{a'}{a} \quad \stackrel{\text{def}}{=} \quad a \,\mathcal{R}\, a'$$

Then

$$\begin{aligned} \Phi_{\mathbf{R}}(S) &= \{\, a \in A \mid (\exists a')\,(\tfrac{a'}{a} \wedge \{\, a' \,\} \subseteq S)\,\} \\ &= \{\, a \in A \mid (\exists a' \in S)\,(a \,\mathcal{R}\, a')\,\} \end{aligned}$$

Thus $\mu\Phi_{\mathbf{R}} = \varnothing$ and

$$\nu\Phi_{\mathbf{R}} = \{\, a \mid (\exists a_i \in A)\,(a \,\mathcal{R}\, a_0 \,\mathcal{R}\, a_1 \,\mathcal{R}\, \ldots)\,\}$$

as $S \subseteq \Phi_{\mathbf{R}}(S) \iff (\forall a \in S)\,((\exists a' \in S)(a \,\mathcal{R}\, a'))$.

What's Next?

- It is useful to have some notation to deal with the "forward" and "back" tracking of rules.
- Closed sets are ones in which we can always track data forwards through rules.
- Dense sets are ones in which we can always track data backwards through rules.
- Recall the Principle of Mathematical Induction . . .
- (co)inductive sets have useful reasoning principles which we outline.

Closed and Dense Sets

- A subset $S \subseteq X$ is closed under a set of rules $\mathbf{R}$ if it is a pre-fixed point of $\Phi_{\mathbf{R}}$, that is

$$\{\, x \in X \mid \exists \frac{S'}{x} \in \mathbf{R} \wedge S' \subseteq S \,\} \stackrel{\text{def}}{=} \Phi_{\mathbf{R}}(S) \subseteq S$$

- S is closed under rule $\frac{H}{c} \in \mathbf{R}$ if

$$H \subseteq S \Longrightarrow c \in S \qquad (*)$$

- Note S is closed under $\mathbf{R}$ just in case it is closed under each rule in $\mathbf{R}$. *Exercise*!

- For each $h \in H$, the assumption $h \in S$ is called an inductive hypothesis.

- A subset $S \subseteq X$ is dense under a set of rules $\mathbf{R}$ if it is a post-fixed point of $\Phi_{\mathbf{R}}$. This means

$$S \subseteq \Phi_{\mathbf{R}}(S) \stackrel{\text{def}}{=} \{\, x \in X \mid \exists \frac{S'}{x} \in \mathbf{R} \wedge S' \subseteq S \,\}$$

- The dense sets for the (previous) rules over $\mathbb{Z}$ are

$$\begin{aligned} S_p &\stackrel{\text{def}}{=} \{p, p-1, p-2, \ldots\} \\ S_n &\stackrel{\text{def}}{=} \{n, n-1, n-2, \ldots\} \\ S_{n,0} &\stackrel{\text{def}}{=} S_n \cup \{0\} \\ S_0 &\stackrel{\text{def}}{=} \{0\} \end{aligned}$$

where $p \geq 1$ and $n \leq -1$. (*Exercise*: closed sets?)

Principle of Induction

Suppose that $I \subseteq X$ is inductively defined, and that $S \subseteq I$. Then

$$S = I \Longleftarrow \begin{cases} \Phi(S) \subseteq S & [S \text{ closed}] \\ \text{or} & \\ \Phi_{\mathbf{R}}(S) \subseteq S & [S \text{ closed under } \mathbf{R}] \end{cases}$$

This follows immediately from the definitions. I is the least prefixed point, that is, least closed set, so $I \subseteq S$.

Principle of Induction – Restated

Suppose that $I \subseteq X$ is inductively defined by $\mathbf{\Phi}$ or $\mathbf{R}$, and that $\phi(i)$ is a predicate on $i \in I$. Then

$$(\forall i)(i \in I \Longrightarrow \phi(i)) \quad \Longleftarrow \quad (\forall \frac{H}{c} \in \mathbf{R}) \; (\underbrace{(\wedge_{h \in H} \phi(h)) \Longrightarrow \phi(c)}_{\substack{\textit{closed under each rule} \\ H \subseteq S \Longrightarrow c \in S}})$$

We may write statements such as "$i \in I \Longrightarrow \phi(i)$ can be proved by induction over $i \in I$".

Principle of Coinduction

Suppose that $C \subseteq X$ is coinductively defined by $\mathbf{\Phi}$ or $\mathbf{R}$. Then

$$x \in C \Longleftarrow \begin{cases} (\exists S)\ (x \in S \land S \subseteq \mathbf{\Phi}(S)) & [S \text{ dense}] \\ \text{or} & \\ (\exists S)\ (x \in S \land S \subseteq \mathbf{\Phi}_{\mathbf{R}}(S)) & [S \text{ dense under } \mathbf{R}] \end{cases}$$

This follows immediately from the definitions. C is the greatest postfixed point, that is, greatest dense set, so $S \subset C$.

What's Next?

- Apparently, we have some "reasoning principles".
- We show that coinduction gives rise to a "method" for showing that two processes are, in some sense, "equivalent".
- In particular, we will discuss under what circumstances a coinductive definition gives rise to an equivalence or even equality.
- We give a small example: We define a model of lazy streams and show that two expressions are in fact equal.

Coinductive Preorders and Equivalence Relations

- Recall: $x \in C \subseteq X \Longleftarrow (\exists S \subseteq X)(x \in S \wedge \underbrace{S \subseteq \Phi(S)}_{\textit{dense}})$ where $C \stackrel{\text{def}}{=} \nu\Phi$ for monotone $\Phi\colon \mathcal{P}(X) \to \mathcal{P}(X)$.

- Suppose $X \stackrel{\text{def}}{=} \mathit{Exp} \times \mathit{Exp}$ is a set of pairs of program or process expressions. We consider the instance

$$(x,x') \in \approx \subseteq \mathit{Exp} \times \mathit{Exp}$$

$$\Longleftarrow (\exists\, \mathcal{B} \subseteq \mathit{Exp} \times \mathit{Exp})((x,x') \in \mathcal{B} \wedge \mathcal{B} \subseteq \Phi(\mathcal{B}))$$

- Question: When is $\approx \stackrel{\text{def}}{=} \nu\Phi$ an equivalence relation? In such a case, we call a dense set $\mathcal{B}$ a bisimulation, and, rephrasing,

$$x \approx x' \quad \Longleftarrow \quad x\,\mathcal{B}\,x' \quad \wedge \quad \underbrace{\mathcal{B}\ \text{is a bisimulation}}_{\mathcal{B}\subseteq\Phi(\mathcal{B})}$$

- Question: When is $\preccurlyeq \stackrel{\text{def}}{=} \nu\Phi$ a preorder? In such a case, we call a dense set $\mathcal{S}$ a simulation, and

$$x \preccurlyeq x' \quad \Longleftarrow \quad x\,\mathcal{S}\,x' \quad \wedge \quad \underbrace{\mathcal{S}\ \text{is a simulation}}_{\mathcal{S}\subseteq\Phi(\mathcal{S})}$$

- Let Eq be the equality relation on Exp. Then $\preccurlyeq \stackrel{\text{def}}{=} \nu\Phi$ is a preorder just in case for each $\mathcal{R}, \mathcal{R}' \subseteq Exp \times Exp$

  - $\Phi(Eq) = Eq$
  - $\Phi(\mathcal{R}) \circ \Phi(\mathcal{R}') \subseteq \Phi(\mathcal{R} \circ \mathcal{R}')$

  Such Φ are called pre-extensional.

- And $\approx \stackrel{\text{def}}{=} \nu\Phi$ is an equivalence relation just in case Φ is pre-extensional, and

  - $\Phi(\mathcal{R})^{op} \subseteq \Phi(R^{op})$

  Such Φ are called extensional. If (additionally) $\approx$ is actually Eq, then Φ is called fully-extensional.

A Model of Streams

- Let L be the "greatest" set such that $L \cong 1 + \mathbb{N} \times L \ldots$ that is, the final coalgebra $\nu\Psi$ for the set endofunctor $\Psi(\xi) = 1 + \mathbb{N} \times \xi$.

- So L is the (unique, up to bijection) set such that for any function $f\colon S \to 1 + \mathbb{N} \times S$, there is $\overline{f}$ with

$$\begin{tikzcd} S \arrow[r, "f"] \arrow[d, "\overline{f}"'] & 1 + \mathbb{N} \times S \arrow[d, "id_1 + id_{\mathbb{N}} \times \overline{f}"] \\ L \arrow[r, "\cong"] & 1 + \mathbb{N} \times L \end{tikzcd}$$

- Key point: $L = \biguplus_{i \leq \omega} \mathbb{N}^i$ is the set of all finite and infinite lists (tuples) of natural numbers, denoted: nil, $n_1 : \mathsf{nil}$, $n_1 : n_2 : \mathsf{nil} \ldots$

- Informally: the isomorphism maps $* \in 1$ to $\mathsf{nil} \in \mathbb{N}^0$, and maps

$$(m, l) \in \mathbb{N} \times \mathbb{N}^i \quad \text{to} \quad m : l \in \mathbb{N}^{i+1}$$

- Given $l \in L$ and $p \geq 1$, write $l_p \in \mathbb{N}$ for the pth element (projection) if it exists. For example,

$$(2 : 5 : 7 : \mathsf{nil})_2 = 5 \qquad (5 : 7 : \mathsf{nil})_{666} \qquad (5 : 7 : \mathsf{nil})_3 \ \text{ both undefined}$$

- Write $l_p \asymp l'_p$ for Kleene equality; then

$$l = l' \stackrel{\text{def}}{=} (l = l' = \mathsf{nil}) \ \vee \ (\forall p)(l_p \asymp l'_p)$$

- In fact (*Exercise*: induction on $m \in \mathbb{N}$)

$$l = l' \Longleftrightarrow (l = l' = \mathsf{nil}) \ \vee \ (\forall m \geq 1)(\forall 1 \geq p \leq m)(l_p \asymp l'_p)$$

- Consider

$$\Phi \colon \mathcal{P}(L \times L) \longrightarrow \mathcal{P}(L \times L)$$

where

$$\Phi(\mathcal{B}) \stackrel{\text{def}}{=} \{\, (l, l') \;\mid\; \begin{cases} l = l' = \mathsf{nil} \\ \vee \\ (\exists h, t, t')(l = h : t \wedge l' = h : t' \wedge t \;\mathcal{B}\; t') \,\} \end{cases}$$

- In fact Φ can be constructed algorithmically from Ψ, with a final coalgebra giving rise to a principle of coinduction, but that is another story ...

- In fact Φ is fully-extensional, ie $\nu\Phi = Eq_L$.

- Extensionality is routine (*Exercise*). For fully-extensional, note $Eq_L = \Phi(Eq_L)$, so $Eq_L \subseteq \Phi(Eq_L)$, hence

$$Eq_L \subseteq \nu\Phi$$

Note that $\nu\Phi \subseteq Eq_L$ if $l \mathrel{\mathcal{B}} l' \Longrightarrow l = l'$. The latter holds as

$$l \mathrel{\mathcal{B}} l' \Longrightarrow (l = l' = \mathsf{nil}) \lor (\forall m)(\forall 1 \geq p \leq m)(l_p \asymp l'_p)$$

provable by induction on $m \in \mathbb{N}$.

- Thus $l = l'$ provided we can find a bisimulation $\mathcal{B}$ with $l \mathrel{\mathcal{B}} l'$, and moreover $\nu\Phi = Eq_L$.

- (Informally/curried) define $\mathsf{M} : (\mathbb{N} \to \mathbb{N}) \times L \longrightarrow L$ by

$$\begin{aligned} \mathsf{M}\, f\, \mathsf{nil} &= \mathsf{nil} \\ \mathsf{M}\, f\, (h : t) &= (f\, h) : (\mathsf{M}\, f\, t) \\ \mathsf{I}\, f\, n &= n : (\mathsf{I}\, f\, (f\, n)) \end{aligned}$$

- Then

$$\mathsf{M}\, f\, (\mathsf{I}\, f\, n) = \mathsf{I}\, f\, (f\, n)$$

if there's a bisimulation $\mathcal{B}$ relating the two operands . . .

such as

$$\mathcal{B} \stackrel{\text{def}}{=} \{\ (\mathsf{M}\,f\,(\mathsf{I}\,f\,n), \mathsf{I}\,f\,(f\,n))\ \mid\ f{:}\mathbb{N} \to \mathbb{N}, n \in \mathbb{N}\ \}$$

$$\begin{array}{ccc}
\mathsf{M}\,f\,(\mathsf{I}\,f\,n) & \mathcal{B} & \mathsf{I}\,f\,(f\,n) \\
\| & & \| \\
f\,n : \mathsf{M}\,f\,(\mathsf{I}\,f\,(f\,n)) & & f\,n : \mathsf{I}\,f\,(f\,(f\,n))
\end{array}$$

- (Informally/curried) define $\mathsf{M}{:}L \times L \longrightarrow L$ by

$$\begin{array}{rclrcl} \mathsf{M}\,l\,\mathsf{nil} & = & l & \mathsf{O}\,\mathsf{nil} & = & \mathsf{nil} \\ \mathsf{M}\,\mathsf{nil}\,l & = & l & \mathsf{O}\,(h:\mathsf{nil}) & = & h:\mathsf{nil} \\ \mathsf{M}\,(h:t)\,(h':t') & = & h:(\mathsf{M}\,(h':t')\,t) & \mathsf{O}\,(h:h':t) & = & h:(\mathsf{O}\,t) \end{array}$$

$$\mathsf{E}\,l = \mathsf{O}\,(\mathsf{tl}(l))$$

- Then

$$\mathsf{M}\,(\mathsf{O}\,l)\,(\mathsf{E}\,l) = l$$

if there's a bisimulation $\mathcal{B}$ relating the two operands ...

such as

$$\mathcal{B} \stackrel{\text{def}}{=} \{\, (\mathsf{M}\,(\mathsf{O}\,l)\,(\mathsf{E}\,l), l) \mid l \in L \,\}$$

$$\begin{array}{ccc}
\mathsf{M}\,(\mathsf{O}\,(h : h' : t))\,(\mathsf{E}\,(h : h' : t)) & \mathcal{B} & h : h' : t \\
\| & & \| \\
\mathsf{M}\,(h : \mathsf{O}\,t)\,(\mathsf{E}\,(h : h' : t)) & & h : h' : t \\
\| & & \| \\
h : \mathsf{M}\,(\mathsf{E}\,(h : h' : t))\,(\mathsf{O}\,t) & & h : h' : t \\
\| & & \| \\
h : \mathsf{M}\,(\mathsf{O}\,(h' : t))\,(\mathsf{E}\,(h' : t)) & & h : h' : t
\end{array}$$

Exercise: what about the other cases?

Overview Part II

- We illustrate coinductive equivalences for a small functional programming language.
- Many of the techniques and ideas which we meet all arise in foundational work on security.
- The vehicle of a functional language is hopefully familiar to you.
- We will define contextual equivalence and bisimilarity, two kinds of equivalence.

Overview Part II - Continued

- The former is intuitively appealing, but the latter is easier to reason about (coinductively).
- Fortunately, they are the same thing (in this setting!).
- We will show this, and give some example applications.
- In more detail, we shall:

Overview Part II - Continued

- Inductively define programs $\underline{4}+\underline{3}$, $\mathsf{hd}(\mathsf{tl}(\underline{5}:\underline{4}:\mathsf{nil}))$ and $\mathsf{F}\,n \equiv \mathsf{if}\ n=\underline{1}\ \mathsf{then}\ \underline{1}\ \mathsf{else}\ n*(f\,(n-\underline{1}))$.
- Inductively define program transitions $P \rightsquigarrow P'$ such as $\mathsf{F}\,\underline{4} \rightsquigarrow^* \underline{4}*(\underline{4}-\underline{1})*(\underline{4}-\underline{1}-\underline{1})*\underline{1} \rightsquigarrow^* \underline{24}$.
- Coinductively define divergence $P\Uparrow$, where this means $P \rightsquigarrow P_1 \rightsquigarrow P_2 \rightsquigarrow P_3 \rightsquigarrow \dots$
- Coinductively define notions of program equivalences such as $\mathsf{F}\,(x*\underline{1}*\underline{7}) \approx \mathsf{F}\,(x*\underline{7})$, and show them all equal.

What's Next?

- During the next few slides we specify a functional language.
- We
  - give types and expressions;
  - a reduction relation (operational semantics); and
  - discuss convergence to values (canonical forms) and divergence of programs.

Types and Expressions

- The types are (the syntax trees) given inductively by

$$\gamma ::= \mathsf{int} \mid \mathsf{bool} \qquad \sigma \quad ::= \gamma \mid [\sigma] \mid \sigma \to \sigma'$$

where $[\sigma]$ is a list type, and $\sigma \to \sigma'$ is a function type.

- The expressions are syntax trees, defined from fixed sets *Var* of variables $x, y, z, v \ldots$, and *Fid* of function identifiers, $\mathsf{F}, \mathsf{G}, \mathsf{M} \ldots$

- If E is an expression in which x_i possibly occurs, where $1 \leq i \leq n$, then

$$E[E_1 \ldots E_i \ldots E_n / x_1 \ldots x_i \ldots x_n] \qquad (x+y)[y, \underline{2}/x, y] = y + \underline{2}$$

is the expression where each E_i simultaneously replaces each x_i.

E	$::=$	x	variable
	$\mid$	$\underline{c} \quad c \in \mathbb{Z} \cup \mathbb{B}$	integer or Boolean constant
	$\mid$	$E_1 \ op \ E_2$	operator on "integers"
	$\mid$	nil_σ	(type indexed) empty list
	$\mid$	$E_1 : E_2$	cons for lists
	$\mid$	$\mathsf{hd}(E) \quad \mathsf{tl}(E)$	head and tail of list
	$\mid$	$\mathsf{elist}(E)$	Boolean test for empty list
	$\mid$	F	function identifier
	$\mid$	$E_1 \ E_2$	function application
	$\mid$	$\mathsf{if}\ E_1\ \mathsf{then}\ E_2\ \mathsf{else}\ E_3$	conditional

A Type Assignment System

- We aim to define type assignments of the form

$$l :: [\mathsf{int}], x :: \mathsf{int} \vdash x + \mathsf{hd}(l) :: \mathsf{int} \qquad \underbrace{x_1 :: \sigma_1, \ldots, x_n :: \sigma_n}_{\Gamma} \vdash E :: \sigma$$

where an environment Γ is a finite partial function from variables to types.

- These are defined parametrically over a set of typed function identifiers. An identifier type takes the form

$$\sigma_1 \to \sigma_2 \to \sigma_3 \to \ldots \to \sigma_a \to \sigma$$

where $a \geq 0$ and σ is not a function type.

- An identifier environment is a finite partial function from identifiers to types, written

$$\Delta = \mathsf{F}_1 :: \iota_1, \ldots, \mathsf{F}_m :: \iota_m.$$

- Given any Δ, we can inductively define our type assignment relation $\Gamma \vdash E :: \sigma$ by a set of rules.

- Write $\mathit{Exp}_\sigma(\Gamma)$ for the set of expressions E with type σ in environment Γ. Write Exp_σ for $\mathit{Exp}_\sigma(\varnothing)$.

- A program expression P is an expression with no occurrences of variables. Call P a program of type σ if $P \in \mathit{Exp}_\sigma$. N.B. P, Q, R range over $\mathit{Prog} \stackrel{\text{def}}{=} \biguplus_\sigma \mathit{Exp}_\sigma$.

$$\frac{\Gamma(x) = \sigma}{\Gamma \vdash x :: \sigma} :: \text{VAR} \qquad \frac{}{\Gamma \vdash \underline{c} :: \gamma} :: \text{CST}$$

$$\frac{\Gamma \vdash E_1 :: \mathsf{int} \quad \Gamma \vdash E_2 :: \mathsf{int}}{\Gamma \vdash E_1 \; op \; E_2 :: \gamma} :: \text{OP} \qquad op \in \{+, *, \leq, \ldots\}$$

$$\frac{}{\Gamma \vdash \mathsf{nil}_\sigma :: [\sigma]} :: \text{NIL} \qquad \frac{\Gamma \vdash E_1 :: \sigma \quad \Gamma \vdash E_2 :: [\sigma]}{\Gamma \vdash E_1 : E_2 :: [\sigma]} :: \text{CONS}$$

$$\frac{\Gamma \vdash E :: [\sigma]}{\Gamma \vdash \mathsf{hd}(E) :: \sigma} :: \text{HD} \quad \frac{\Gamma \vdash E :: [\sigma]}{\Gamma \vdash \mathsf{tl}(E) :: [\sigma]} :: \text{TL} \quad \frac{\Gamma \vdash E :: [\sigma]}{\Gamma \vdash \mathsf{elist}(E) :: \mathsf{bool}} :: \text{ELIST}$$

$$\frac{\Delta(\mathsf{F}) = \iota}{\Gamma \vdash \mathsf{F} :: \iota} :: \text{IDR}$$

$$\frac{\Gamma \vdash E_1 :: \sigma_2 \to \sigma_1 \quad \Gamma \vdash E_2 :: \sigma_2}{\Gamma \vdash E_1\, E_2 :: \sigma_1} :: \text{AP}$$

$$\frac{\Gamma \vdash E_1 :: \mathsf{bool} \quad \Gamma \vdash E_2 :: \sigma \quad \Gamma \vdash E_3 :: \sigma}{\Gamma \vdash \mathsf{if}\ E_1\ \mathsf{then}\ E_2\ \mathsf{else}\ E_3 :: \sigma} :: \text{COND}$$

We write $P :: \sigma$ for $\varnothing \vdash P :: \sigma$

Function Declarations

- To define run-time execution, given some Δ, we first declare the meanings of function identifiers.

- For example,

$$\mathsf{I}\, x\, y \equiv x + y \qquad \mathsf{F}\, x \equiv \mathsf{if}\ x \leq 1\ \mathsf{then}\ 1\ \mathsf{else}\ x * \mathsf{F}\,(x-1)$$

- In general, declare

$$\mathsf{I}\, x_1\, x_2 \ldots x_a \equiv D_{\mathsf{I}}$$

for each identifier $\mathsf{I} :: \sigma_1 \to \sigma_2 \to \sigma_3 \to \ldots \to \sigma_a \to \sigma$ where

$$x_1 :: \sigma_1 \ldots x_a :: \sigma_a \vdash D_{\mathsf{I}} :: \sigma$$

A Small Step Reduction Relation

$$
\begin{array}{rcl}
\mathsf{I}\,\underline{2}\,\underline{3} & \leadsto & (x+y)[\underline{2},\underline{3}/x,y] = \underline{2}+\underline{3} \\
 & \leadsto & \underline{5} \not\leadsto \\
\\
\mathsf{F}\,\underline{2} & \leadsto & \mathsf{if}\ \underline{2} \le \underline{1}\ \mathsf{then}\ \underline{1}\ \mathsf{else}\ \underline{2} * \mathsf{F}\,(\underline{2}-\underline{1}) \\
 & \leadsto & \mathsf{if}\ \underline{f}\ \mathsf{then}\ \underline{1}\ \mathsf{else}\ \underline{2} * \mathsf{F}\,(\underline{2}-\underline{1}) \\
 & \leadsto & \underline{2} * \mathsf{F}\,(\underline{2}-\underline{1}) \\
 & \leadsto & \underline{2} * \mathsf{if}\ (\underline{2}-\underline{1}) \le \underline{1}\ \mathsf{then}\ \underline{1}\ \mathsf{else}\ \underline{2} * \mathsf{F}\,((\underline{2}-\underline{1})-\underline{1}) \\
\text{(Reflexive, transitive closure)} & \leadsto^* & \underline{2} \not\leadsto \\
\\
\mathsf{tl}(\underline{4} : (\underline{2}-\underline{1}) : \mathsf{nil}) & \leadsto & (\underline{2}-\underline{1}) : \mathsf{nil} \not\leadsto
\end{array}
$$

- The evaluation contexts are defined by

$$\mathcal{E} ::= - \mid -\ op\ P \mid \underline{n}\ op\ - \mid \mathsf{hd}(-) \mid \mathsf{tl}(-) \mid \mathsf{elist}(-) \mid -\ P \mid \mathsf{if}\ -\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2$$

- and the reduction relation $\rightsquigarrow\ \subseteq \biguplus_\sigma (Exp_\sigma \times Exp_\sigma)$ by

$$\frac{P \rightsquigarrow P'}{\mathcal{E}[P] \rightsquigarrow \mathcal{E}[P']} \qquad \underline{n}\ op\ \underline{m} \rightsquigarrow \underline{n\ op\ m} \qquad \mathsf{hd}(P : P') \rightsquigarrow P$$

$$\mathsf{tl}(P : P') \rightsquigarrow P' \qquad \mathsf{hd}(\mathsf{nil}) \rightsquigarrow \mathsf{hd}(\mathsf{nil}) \qquad \mathsf{tl}(\mathsf{nil}) \rightsquigarrow \mathsf{tl}(\mathsf{nil})$$

$$\mathsf{elist}(P : P') \rightsquigarrow \underline{f} \qquad \mathsf{elist}(\mathsf{nil}) \rightsquigarrow \underline{t}$$

$$\mathsf{F}\, P_1 \ldots P_a \rightsquigarrow D_{\mathsf{F}}[P_1, \ldots, P_a / x_1, \ldots, x_a]$$

$$\mathsf{if}\ \underline{t}\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 \rightsquigarrow P_1 \qquad \mathsf{if}\ \underline{f}\ \mathsf{then}\ P_1\ \mathsf{else}\ P_2 \rightsquigarrow P_2$$

Convergence and Divergence

- Define

$$P \leadsto \stackrel{\text{def}}{=} (\exists R)(P \leadsto R) \qquad\qquad P \not\leadsto \stackrel{\text{def}}{=} (\nexists R)(P \leadsto R)$$

- Note that $\mathsf{I}\,\underline{2}\,\underline{3} \leadsto^* \underline{5} \not\leadsto$. The program converges.
- If $\mathsf{G}\,z \equiv \mathsf{G}\,(z+\underline{2})$ then $G\,\underline{0} \leadsto \mathsf{G}(\underline{0}+\underline{2}) \leadsto^{\omega}$. The program diverges.
- We define, for $P, P' \in Prog$,

$$\begin{aligned} P \Downarrow &\stackrel{\text{def}}{=} (\exists P')(P \leadsto^* P' \wedge (P' \not\leadsto)) \\ P \Uparrow &\stackrel{\text{def}}{=} (\forall P')(P \leadsto^* P' \Longrightarrow (P' \leadsto)) \end{aligned}$$

- Reduction is deterministic. *Exercise*: Induction over $\leadsto$.

Values

- In fact if $P \Downarrow$, then $(\exists V)(P \rightsquigarrow^{*} V)$, where values V are defined as programs in *Prog* such that

$$V ::= \underline{c} \mid \mathsf{nil} \mid P : P' \mid \mathsf{F}\, \underbrace{P_1 \ldots P_l}_{l<a}$$

- Idea: a value is a “fully reduced” program.
- Lists are lazy: reduce elements only if extracted by head or tail.
- Only reduce “identifier applications” if identifier has all its a arguments.

- We can show that for $P \in Prog$,

$$P \text{ is a value} \quad \Longleftrightarrow \quad P \not\rightsquigarrow$$

- $\Longrightarrow$ is trivial. $\Longleftarrow$ by induction on type assignments:

$$\Gamma \vdash E :: \sigma \quad \Longrightarrow \quad ((\Gamma = \emptyset \wedge E \not\rightsquigarrow) \Longrightarrow E \text{ is a value })$$

- Then it is immediate that

$$P \Downarrow \quad \Longleftrightarrow \quad (\exists V)(P \rightsquigarrow^* V)$$

Divergence Coinductively

$$\frac{P\uparrow}{P\ op\ Q\uparrow} \qquad \frac{Q\uparrow}{P\ op\ Q\uparrow}\ P \rightsquigarrow^* \underline{n} \qquad \frac{P\uparrow}{\mathsf{hd}(P)\uparrow} \qquad \frac{P\uparrow}{\mathsf{tl}(P)\uparrow} \qquad \frac{P\uparrow}{\mathsf{elist}(P)\uparrow}$$

$$\frac{P\uparrow}{P\,Q\uparrow} \qquad \frac{D_{\mathsf{F}}[\vec{P}, Q/\vec{x}, x]\uparrow}{P\,Q\uparrow}\ P \rightsquigarrow^* \mathsf{F}\,P_1 \ldots P_{a-1} \qquad \frac{P\uparrow}{\mathsf{if}\ P\ \mathsf{then}\ Q\ \mathsf{else}\ Q'\uparrow}$$

$$\frac{Q\uparrow}{\mathsf{if}\ P\ \mathsf{then}\ Q\ \mathsf{else}\ Q'\uparrow}\ P \rightsquigarrow^* \underline{t} \qquad \frac{Q'\uparrow}{\mathsf{if}\ P\ \mathsf{then}\ Q\ \mathsf{else}\ Q'\uparrow}\ P \rightsquigarrow^* \underline{f}$$

- The set $\uparrow \subseteq \mathit{Prog}$ is coinductively defined by these rules.
- We can show that for all $P \in \mathit{Prog}$,

$$P \Uparrow \quad \iff \quad P \uparrow$$

- For example, if $\mathsf{F}\, x \equiv \mathsf{F}\, x$, then $\mathsf{F}\, \underline{0} \uparrow$ provided $\mathsf{F}\, \underline{0} \in D$ for some dense set D. Can take $D \stackrel{\text{def}}{=} \{\mathsf{F}\, \underline{0}\}$!

Where Now?

- We define formally contextual equivalence.
- Two programs are equivalent, if, when placed in any "larger" program, the resulting programs both converge.
- It is difficult to establish such equivalences: how do you check this for all larger programs?
- *Problem is circumvented by defining bisimilarity, another equivalence, which is more tractable … but*
- *which coincides with contextual equivalence.*
- To show this, the trick is to show bisimilarity a congruence …

Contextual Preorder

- $$\Gamma \vdash E \mathcal{R} E' : \sigma \quad \stackrel{\text{def}}{=} \quad (E, E') \in \mathcal{R} \subseteq \uplus_{\Gamma,\sigma}(Exp_\sigma(\Gamma) \times Exp_\sigma(\Gamma))$$

- The contextual preorder

$$x_1 :: \sigma_1, \ldots, x_n :: \sigma_n \vdash E \leq E' :: \sigma$$

means: for all "contexts" $v :: \sigma \vdash C :: \tau$, and programs $P_i :: \sigma_i$,

$$C[E[\vec{P}/\vec{x}]/v] \Downarrow \quad \Longrightarrow \quad C[E'[\vec{P}/\vec{x}]/v] \Downarrow$$

- Contextual equality $\Gamma \vdash E \doteq E' :: \sigma$ is the symmetrization of the contextual preorder, which is a preorder. If the environment is empty, we write $P \doteq_\sigma Q$.

Proving Contextual Equality

- We would expect $\underline{2} + \underline{3} \doteq_{\mathsf{int}} \underline{5}$. To show this, need to prove convergence of $\underline{2} + \underline{3}$ in all contexts implies convergence of $\underline{5}$ in all contexts.
- We would expect $\mathsf{F}\,P \doteq_{\sigma} D_{\mathsf{F}}[P/v]$ when $\mathsf{F}\,x \equiv D_{\mathsf{F}}$.
- *Exercise*: Try proving these facts by induction over all contexts.
- The quantification over all contexts makes establishing these facts tricky.
- As Nat West Bank would say: there is a better way …

Borrowing from Process Algebra

- In process algebra, two processes are equivalent if any transition performed by one can be performed by the other, and the resulting processes are also equivalent.
- We define a concept of “transition” for our functional language.
- Key Idea: Transitions can indicate what can be observed of programs P, once “fully evaluated”.
- Any program can perform a transition α if it first converges (to V) and V can perform α.

- We will have

$$
\begin{tikzcd}
\underline{2}+\underline{3} \arrow[d, "\rightsquigarrow"'] & = & \underline{2}+\underline{3} \arrow[dd, "5"'] & \underline{5} \arrow[dd, "5"'] & \mathsf{tl}(\underline{3} : \underline{2} : \mathsf{nil}) \arrow[dd, "\mathsf{hd}"'] & P \arrow[d, "\rightsquigarrow^{*}"'] \\
\underline{5} \arrow[d, "5"'] & & & & & V \arrow[d, "\alpha"'] \\
\top & & \top & \top & \underline{2} & \top
\end{tikzcd}
$$

A Transition Relation

- Actions $\alpha \in Act$ are given by

$$\alpha ::= c \mid \mathsf{nil} \mid \mathsf{hd} \mid \mathsf{tl} \mid \mathsf{elist} \mid @P$$

- and transition relationships

$$P \xrightarrow{\alpha} \xi \in Prog \times Act \times (Prog \cup \{\top\}) \qquad \text{by}$$

$$\mathsf{F}\vec{P} \xrightarrow{@Q} \mathsf{F}\vec{P}Q \qquad \underline{c} \xrightarrow{c} \top$$

$$\mathsf{nil} \xrightarrow{\mathsf{nil}} \top \qquad P : P' \xrightarrow{\mathsf{hd}} P \qquad P : P' \xrightarrow{\mathsf{tl}} P'$$

$$P : P' \xrightarrow{\mathsf{elist}} \underline{f} \qquad \mathsf{nil} \xrightarrow{\mathsf{elist}} \underline{t} \qquad \frac{P \rightsquigarrow P' \quad P' \xrightarrow{\alpha} \xi}{P \xrightarrow{\alpha} \xi}$$

Results about Convergence and Transitions

- Recall that $P\Downarrow \iff (\exists V)\ (P \rightsquigarrow^* V)$. Then
- $$P \xrightarrow{\alpha} \xi \implies P\Downarrow$$

  can be proved by rule induction over transitions; and
- $$P\Downarrow \implies (\exists\alpha)(\exists\xi)(\exists V)(P \xrightarrow{\alpha} \xi \quad \wedge \quad P \rightsquigarrow^* V \xrightarrow{\alpha} \xi)$$

  follows trivially (using also the single inductive rule).
- Each $\xrightarrow{\alpha}$ is a partial function, by induction on transitions.

Similarity and Bisimilarity

We can define a coinductive notion of similarity,

$$\preccurlyeq \stackrel{\text{def}}{=} \nu\Phi \in \mathcal{P}(\biguplus_{\sigma}(Exp_\sigma \times Exp_\sigma))$$

$$\Phi \colon \mathcal{P}(\biguplus_{\sigma}(Exp_\sigma \times Exp_\sigma)) \longrightarrow \mathcal{P}(\biguplus_{\sigma}(Exp_\sigma \times Exp_\sigma))$$

where

$$\Phi(\mathcal{S}) \stackrel{\text{def}}{=} \left\{ (P,Q) \;\middle|\; \begin{array}{l} \forall P \xrightarrow{\alpha} P' \Longrightarrow Q \xrightarrow{\alpha} Q' \wedge P' \mathrel{\mathcal{S}} Q' \\ \vee \\ P \xrightarrow{\alpha} \top \Longrightarrow Q \xrightarrow{\alpha} \top \end{array} \right\}$$

Bisimilarity is the set $\approx \stackrel{\text{def}}{=} \nu\Phi$ where

$$\Phi(\mathcal{B}) \stackrel{\text{def}}{=} \left\{ (P,Q) \;\middle|\; \begin{array}{l} (\forall\alpha,P',Q')(P \xrightarrow{\alpha} P' \Longrightarrow Q \xrightarrow{\alpha} Q' \wedge P' \mathcal{B} Q') \\ \qquad\quad \wedge \\ (\forall\alpha,P',Q')(Q \xrightarrow{\alpha} Q' \Longrightarrow P \xrightarrow{\alpha} P' \wedge P' \mathcal{B} Q') \\ \vee \\ P \xrightarrow{\alpha} \top \Longleftrightarrow Q \xrightarrow{\alpha} \top \end{array} \right\}$$

Exercise: show Φ is extensional, and hence that bisimilarity is an equivalence relation (and similarity is a preorder).

Open Similarity and Bisimilarity

- Suppose given $\mathcal{R} \subseteq \biguplus_\sigma (Exp_\sigma \times Exp_\sigma)$.
- We write $\Gamma \vdash E \; \mathcal{R}^\circ \; E' : \sigma$ just in case

$$(\forall P_i \in Exp_{\sigma_i}) \;\; (\varnothing \vdash E[\vec{P}/\vec{x}] \; \mathcal{R} \; E'[\vec{P}/\vec{x}] : \sigma)$$

where the types σ_i are those appearing in Γ.

- Call these relationships the open extension of $\mathcal{R}$.
- We obtain open similarity and open bisimilarity, subsets of $\biguplus_{\Gamma,\sigma}(Exp_\sigma(\Gamma) \times Exp_\sigma(\Gamma))$.

Relating Bisimilarity and Contextual Equivalence

A central theorem is: For all Γ, E, E' and σ,

$$\Gamma \vdash E \precsim^{\circ} E' :: \sigma \iff \Gamma \vdash E \leq E' :: \sigma$$

$$\Gamma \vdash E \approx^{\circ} E' :: \sigma \iff \Gamma \vdash E \doteq E' :: \sigma$$

We can prove this by showing that

- $\precsim$ is a precongruence. $\Longrightarrow$ follows easily from the definitions, plus our Results about Convergence and Transitions
- $\biguplus_{\sigma}\{\,(P,Q) \mid P \leq_{\sigma} Q\,\}$ is a simulation. $\Longleftarrow$ follows from the definitions.

An Example Equivalence

Two facts (proved later):

$$P \leadsto P' \Longrightarrow P \approx P' \quad \text{and} \quad P \approx P' \Longrightarrow C[P/v] \approx C[P'/v]$$

Declare

$$\begin{aligned} \mathsf{M}\, f\, l &\equiv \mathsf{if\ elist}(l)\ \mathsf{then\ nil\ else}\ f\, \mathsf{hd}(l) : \mathsf{M}\, f\, (\mathsf{tl}(l)) \\ \mathsf{I}\, f\, x &\equiv x : \mathsf{I}\, f\, (f\, x) \end{aligned}$$

Then

$$\begin{aligned} & f :: \sigma \to \sigma, x :: \sigma \vdash \mathsf{I}\, f\, (f\, x) \doteq \mathsf{M}\, f (\mathsf{I}\, f\, x) :: [\sigma] \\ \Longleftarrow \quad & f :: \sigma \to \sigma, x :: \sigma \vdash \mathsf{I}\, f\, (f\, x) \approx^{\circ} \mathsf{M}\, f (\mathsf{I}\, f\, x) :: [\sigma] \\ \Longleftarrow \quad & \forall F :: \sigma \to \sigma, P :: \sigma \qquad \mathsf{I}\, F\, (F\, P) \approx_{[\sigma]} \mathsf{M}\, F (\mathsf{I}\, F\, P) \end{aligned}$$

Define

$$\mathcal{B} \stackrel{\text{def}}{=} \biguplus_{\sigma, i \geq 0} \left\{ \begin{array}{c} (\mathsf{I}\,(F\,(F\,(F^i\,P))), \\ \mathsf{M}\,F\,(\mathsf{tl}^i(\mathsf{I}\,F\,P))) \end{array}, (Q, Q') \mid P :: \sigma, F :: \sigma \to \sigma, Q \approx Q' \right\}$$

Then

$$\mathsf{I}(F\,(F\,(F^i\,P))) \rightsquigarrow F\,(F^i\,P) : \mathsf{I}(F\,(F\,(F\,(F^i\,P)))) \begin{array}{l} \xrightarrow{\mathsf{hd}} F\,(F^i\,P) \\ \xrightarrow{\mathsf{tl}} \mathsf{I}\,F\,(F\,(F^{i+1}\,P)) \end{array}$$

$$\mathsf{M}\,F\,(\mathsf{tl}^i(\mathsf{I}\,F\,P)) \rightsquigarrow^* F\,\mathsf{hd}(\mathsf{tl}^i(\mathsf{I}\,F\,P)) : \mathsf{M}\,F\,(\mathsf{tl}^{i+1}(\mathsf{I}\,F\,P)) \begin{array}{l} \xrightarrow{\mathsf{hd}} F\,\mathsf{hd}(\mathsf{tl}^i(\mathsf{I}\,F\,P)) \\ \xrightarrow{\mathsf{tl}} \mathsf{M}\,F\,(\mathsf{tl}^{i+1}(\mathsf{I}\,F\,P)) \end{array}$$

$$\mathsf{hd}(\mathsf{tl}^i(\mathsf{I}\,F\,P)) \rightsquigarrow F^iP \wedge \mathit{FACTS} \implies \mathsf{F}\,\mathsf{hd}(\mathsf{tl}^i(\mathsf{I}\,F\,P)) \approx \mathsf{F}\,(F^i\,P)$$

Where Now?

- The "fact" that for any context $C \in Exp_{\tau}(v :: \sigma)$

$$P \approx P' \Longrightarrow C[P/v] \approx C[P'/v]$$

is not easy to prove.

- The next few slides give a proof of this fact:
  - Define a new relation;
  - show the relation has the substitution property;
  - prove intermediate lemmas relating new relation to reductions and transitions;
  - show new relation equals (open) [bi]similarity.

[Pre]Congruences

Let $\mathcal{R} \subseteq \biguplus_{\Gamma,\sigma}(Exp_\sigma(\Gamma) \times Exp_\sigma(\Gamma))$. Then $\mathcal{R}$ is called a precongruence if

PCrf For any $\Gamma \vdash E :: \sigma$ we have $\Gamma \vdash E \,\mathcal{R}\, E : \sigma$.

PCtr For any $\Gamma \vdash E \,\mathcal{R}\, E' : \sigma$ and $\Gamma \vdash E' \,\mathcal{R}\, E'' : \sigma$ we have $\Gamma \vdash E \,\mathcal{R}\, E'' : \sigma$.

PCwk Weakening of contexts.

PCsb For any relationships $\Gamma \vdash E \,\mathcal{R}\, E' : \sigma$ and $\Gamma, x :: \sigma \vdash T \,\mathcal{R}\, T' : \sigma'$ we have $\Gamma \vdash T[E/x] \,\mathcal{R}\, T'[E'/x] : \sigma'$.

PCsy A congruence satisfies additionally

$$\Gamma \vdash E \,\mathcal{R}\, E' : \sigma \Longrightarrow \Gamma \vdash E' \,\mathcal{R}\, E : \sigma$$

The Howe Relation

- To prove similarity a precongruence, we adopt Howe's method.
- We inductively define $\Gamma \vdash E \precsim^{\bullet} E' :: \sigma$, prove these form a precongruence, and then show

$$\Gamma \vdash E \precsim^{\bullet} E' :: \sigma \Longleftrightarrow \Gamma \vdash E \precsim^{\circ} E' :: \sigma$$

$$\frac{\Gamma, x{:}\sigma \vdash x \preccurlyeq^{\circ} E :: \sigma}{\Gamma, x{:}\sigma \vdash x \preccurlyeq^{\bullet} E :: \sigma} \qquad \frac{\Gamma \vdash \underline{c} \preccurlyeq^{\circ} E :: \gamma}{\Gamma \vdash \underline{c} \preccurlyeq^{\bullet} E :: \gamma}$$

$$\frac{\Gamma \vdash E_1 \preccurlyeq^{\bullet} \hat{E}_1 :: \mathsf{int} \quad \Gamma \vdash E_2 \preccurlyeq^{\bullet} \hat{E}_2 :: \mathsf{int} \quad \Gamma \vdash \hat{E}_1 \; op \; \hat{E}_2 \preccurlyeq^{\circ} T :: \gamma}{\Gamma \vdash E_1 \; op \; E_2 \preccurlyeq^{\bullet} T :: \gamma}$$

$$\frac{\Gamma \vdash \mathsf{nil} \preccurlyeq^{\circ} E :: \gamma}{\Gamma \vdash \mathsf{nil} \preccurlyeq^{\bullet} E :: \gamma}$$

$$\frac{\Gamma \vdash E_1 \preccurlyeq^{\bullet} \hat{E}_1 :: \sigma \quad \Gamma \vdash E_2 \preccurlyeq^{\bullet} \hat{E}_2 :: [\sigma] \quad \Gamma \vdash \hat{E}_1 : \hat{E}_2 \preccurlyeq^{\circ} T :: [\sigma]}{\Gamma \vdash E_1 : E_2 \preccurlyeq^{\bullet} T :: [\sigma]}$$

$$\frac{\Gamma \vdash E \preccurlyeq^{\bullet} \hat{E} :: [\sigma] \quad \Gamma \vdash \Box\hat{E} \preccurlyeq^{\circ} T :: \sigma}{\Gamma \vdash \Box E \preccurlyeq^{\bullet} T :: [\sigma]} [\text{where } \Box \in \{\,\mathsf{hd}, \mathsf{tl}, \mathsf{elist}\,\}]$$

$$\frac{\Gamma \vdash \mathsf{F} \preccurlyeq^{\circ} E :: \iota}{\Gamma \vdash \mathsf{F} \preccurlyeq^{\bullet} E :: \iota}$$

$$\frac{\Gamma \vdash E_1 \preccurlyeq^{\bullet} \hat{E}_1 :: \sigma \to \sigma \quad \Gamma \vdash E_2 \preccurlyeq^{\bullet} \hat{E}_2 :: \sigma \quad \Gamma \vdash \hat{E}_1\,\hat{E}_2 \preccurlyeq^{\circ} T :: \sigma}{\Gamma \vdash E_1\,E_2 \preccurlyeq^{\bullet} T :: \sigma}$$

$$\frac{\begin{array}{l} \Gamma \vdash E_1 \preccurlyeq^{\bullet} \hat{E}_1 :: \mathsf{bool} \\ \Gamma \vdash E_2 \preccurlyeq^{\bullet} \hat{E}_2 :: \sigma \\ \Gamma \vdash E_3 \preccurlyeq^{\bullet} \hat{E}_3 :: \sigma \\ \Gamma \vdash \mathsf{if}\ \hat{E}_1\ \mathsf{then}\ \hat{E}_2\ \mathsf{else}\ \hat{E}_3 \preccurlyeq^{\circ} T :: \sigma' \end{array}}{\Gamma \vdash \mathsf{if}\ E_1\ \mathsf{then}\ E_2\ \mathsf{else}\ E_3 \preccurlyeq^{\bullet} T :: \sigma'}$$

Some General Properties of Howe

Htr For all $\boxed{\Gamma \vdash E \preccurlyeq^{\bullet} E' :: \sigma}$ and $\Gamma \vdash E' \preccurlyeq^{\circ} E'' :: \sigma$ we have $\Gamma \vdash E \preccurlyeq^{\bullet} E'' :: \sigma$.

Hrf For all $\boxed{\Gamma \vdash E :: \sigma}$ we have $\Gamma \vdash E \preccurlyeq^{\bullet} E :: \sigma$.

Hoh For all $\Gamma \vdash E \preccurlyeq^{\circ} E' :: \sigma$ we have $\Gamma \vdash E \preccurlyeq^{\bullet} E' :: \sigma$.

Hsb For all $\boxed{\Gamma, x{:}\sigma \vdash T \preccurlyeq^{\bullet} T' :: \sigma'}$ and $\Gamma \vdash E \preccurlyeq^{\bullet} E' :: \sigma$, we have $\Gamma \vdash T[E/x] \preccurlyeq^{\bullet} T'[E'/x] :: \sigma'$.

Note **Hoh** follows from **Htr** and **Hrf**.

Sketch Proofs

These results follow by induction over the boxed judgments.

Hsb requires

$$\Gamma, x{:}\sigma \vdash T \preccurlyeq^{\circ} T' :: \sigma' \wedge \Gamma \vdash E' :: \sigma \Longrightarrow \\ \Gamma \vdash T[E'/x] \preccurlyeq^{\circ} T'[E'/x] :: \sigma' \quad \dagger$$

(*Exercise*: use definition of open similarity).

Base induction step introducing variables: Suppose that $\Gamma, x{:}\sigma \vdash x \preccurlyeq^{\bullet} T' :: \sigma'$. Then by definition $\Gamma, x{:}\sigma \vdash x \preccurlyeq^{\circ} T' :: \sigma'$. Hence by $\dagger$

$$\Gamma \vdash E' \preccurlyeq^{\circ} T'[E'/x] :: \sigma'$$

and by **Htr** and $\Gamma \vdash E \preccurlyeq^{\bullet} E' :: \sigma$ we are done.

Induction step for applications: Informally!

$$\boxed{\Gamma, v \vdash T \preccurlyeq^{\bullet} T' \wedge \Gamma \vdash E \preccurlyeq^{\bullet} E' \Longrightarrow \Gamma \vdash T[E/v] \preccurlyeq^{\bullet} T'[E'/v]}$$

$$\dfrac{\dfrac{\Gamma \vdash T_1[E/v] \preccurlyeq^{\bullet} \hat{T}_1[E'/v]}{\Gamma, v \vdash T_1 \preccurlyeq^{\bullet} \hat{T}_1} \quad \dfrac{\Gamma \vdash T_2[E/v] \preccurlyeq^{\bullet} \hat{T}_2[E'/v]}{\Gamma, v \vdash T_2 \preccurlyeq^{\bullet} \hat{T}_2} \quad \dfrac{\Gamma \vdash (\hat{T}_1\,\hat{T}_2)[E'/v] \preccurlyeq^{\circ} (T_1\,T_2)[E'/v]}{\Gamma, v \vdash \hat{T}_1\,\hat{T}_2 \preccurlyeq^{\circ} T_1'\,T_2'}}{\Gamma, v \vdash T_1\,T_2 \preccurlyeq^{\bullet} T_1'\,T_2'}$$

$$\textbf{Htr} \quad \Longrightarrow \quad \Gamma, v \vdash (T_1\,T_2)[E/v] \preccurlyeq^{\bullet} (T_1'\,T_2')[E'/v]$$

Some Lemmas

TB If $P \rightsquigarrow P'$ then $P \approx P'$. *Exercise*: Check $\rightsquigarrow \cup Eq_{Exp}$ is a bisimulation. (Follows from Results about Convergence and Transitions.)

ST If $P \preccurlyeq Q$ and $P \rightsquigarrow P'$ then $P' \preccurlyeq Q$. (Ditto!)

HT If $P \preccurlyeq^{\bullet}_{\sigma} Q$ and $P \rightsquigarrow P'$ then $P' \preccurlyeq^{\bullet}_{\sigma} Q$. Proof: induct over transitions.

We illustrate informally one step in proof of **HT**, in case the transition is $\mathsf{F}\,P_1 \rightsquigarrow D_{\mathsf{F}}[P_1/x]$.

We have

$$\dfrac{\dfrac{\mathsf{F} \preccurlyeq_{\sigma\to\sigma'} \hat{F}}{\mathsf{F} \preccurlyeq^{\bullet}_{\sigma\to\sigma'} \hat{F}} \qquad P_1 \preccurlyeq^{\bullet}_{\sigma} \hat{P}_1 \qquad \hat{F}\,\hat{P}_1 \preccurlyeq_{\sigma'} Q}{F\,P_1 \preccurlyeq^{\bullet}_{\sigma'} Q}$$

and thus

$$\begin{tikzcd}
 & & & & \mathsf{F} \arrow[d, swap, "@\,\hat{P}_1"] & \preccurlyeq & \hat{F} \arrow[d, swap, "@\,\hat{P}_1"] & & \\
D_{\mathsf{F}}[P_1/x] & \underbrace{\preccurlyeq^{\bullet}}_{\textbf{Hsb}} & D_{\mathsf{F}}[\hat{P}_1/x] & \underbrace{\preccurlyeq}_{\textbf{TB}} & \mathsf{F}\,\hat{P}_1 & \preccurlyeq & \hat{\mathsf{F}}\,\hat{P}_1 & \preccurlyeq & Q
\end{tikzcd}$$

Similarity and Howe Coincide

$$\Gamma \vdash E \preccurlyeq^{\circ} E' :: \sigma \Longleftrightarrow \Gamma \vdash E \preccurlyeq^{\bullet} E' :: \sigma.$$

$\Longrightarrow$ follows from **Hoh**.

$\Longleftarrow$ follows by showing that $P \preccurlyeq^{\bullet}_{\sigma} Q \Longrightarrow P \preccurlyeq_{\sigma} Q$. This follows (by coinduction) if we can show that

$$\mathcal{S} \stackrel{\text{def}}{=} \biguplus_{\sigma} \{ (P, Q) \mid P \preccurlyeq^{\bullet}_{\sigma} Q \}$$

is a simulation. This follows from the lemmas, using induction on transitions, and Howe properties.

Consider

$$\frac{P \rightsquigarrow P' \qquad P' \xrightarrow{\alpha} \xi}{P \xrightarrow{\alpha} \xi}$$

$$\begin{tikzcd}
P \arrow[r, equal] \arrow[dd, "\alpha"'] & P \arrow[d, "\rightsquigarrow"'] & \preccurlyeq^{\bullet}_{\sigma} & Q & \\
 & P' \arrow[d, "\alpha"'] & \preccurlyeq^{\bullet}_{\sigma} & Q \arrow[d, "\alpha"'] & \textbf{HT} \\
\xi \arrow[r, equal] & \xi & & \xi' & \textbf{INDUCT}
\end{tikzcd}$$

and $\xi \preccurlyeq^{\bullet}_{\sigma} \xi' \vee \xi = \xi' = \top$, also by induction.

Consider

$$P_1 : P_2 \xrightarrow{\mathsf{hd}} P_1 \qquad \text{and} \qquad P_1 : P_2 \preccurlyeq^{\bullet}_{[\sigma]} Q$$

By definition of Howe,

$$(\exists \hat{P}_i \in \mathit{Prog})\ (P_i \preccurlyeq^{\bullet}_{\sigma} \hat{P}_i \wedge \hat{P}_1 : \hat{P}_2 \preccurlyeq_{[\sigma]} Q)$$

and then

$$\begin{array}{ccccc} P_1 : P_2 & \preccurlyeq^{\bullet}_{[\sigma]} & \hat{P}_1 : \hat{P}_2 & \preccurlyeq_{[\sigma]} & Q \\ \mathsf{hd}\downarrow & & \mathsf{hd}\downarrow & & \mathsf{hd}\downarrow \\ P_1 & \preccurlyeq^{\bullet}_{[\sigma]} & \hat{P}_1 & \preccurlyeq_{[\sigma]} & Q' \end{array}$$

Hsb: (**Hrf:** $x : x'$)

Similarity

So $P_1 \preccurlyeq^{\bullet}_{[\sigma]} Q'$ by **Htr**.

Where Now?

- Note: A congruence is a relation preserved by expression constructors (see previous slide) . . .
- We can complete the proof of the coincidence of bisimilarity and contextual equivalence.
- Before doing so, we remark that bisimilarity is the equivalence relation generated by the similarity preorder (used in the proof).
- And finally we look at another equivalence.

Bisimilarity is Generated from Similarity

- Bisimilarity is the equivalence relation generated from similarity.
- This can be proved using the definitions. For example, to show (left disjunct of)

$$P \approx P' \Longrightarrow (P \precsim P' \wedge P' \precsim P)$$

let $\mathcal{S} \stackrel{\text{def}}{=} \{ (P, P') \in \mathit{Prog} \mid P \approx P' \}$ and verify it is a simulation, hence contained in $\precsim$.

Open Similarity is a Precongruence & Open Bisimilarity is a Congruence

Exercise: open bisimilarity is the equivalence relation generated by open similarity. Thus, properties of open (bi)similarity:

SBrf Reflexive because (bi)similarity is reflexive.

SBtr Transitive because(bi)similarity is transitive.

SBwk Weakening is immediate from the definitions.

SBsb The substitution property holds for Howe, hence(!) for open (bi)similarity.

SBsy And open bisimilarity must be symmetric!

Open Similarity implies Contextual Preorder & Open Bisimilarity implies Contextual Equivalence

We have

$$\Gamma \vdash E \precsim^{\circ} E' :: \sigma \Longrightarrow \Gamma \vdash E \leq E' :: \sigma$$

Proof: Let $\Gamma \vdash E \precsim^{\circ} E' :: \sigma$, and suppose $C[E[\vec{P}/\vec{x}]/v] \Downarrow$. Then by **SBsb** we have

$$\varnothing \vdash C[E[\vec{P}/\vec{x}]/v] \precsim^{\circ} C[E'[\vec{P}/\vec{x}]/v] :: \sigma$$

By Results on Convergence and Transitions, convergence "corresponds" to transitions, and hence $C[E'[\vec{P}/\vec{x}]/v] \Downarrow$.

Exercise: Think about result for open bisimilarity.

Contextual Preorder implies Open Similarity & Contextual Equivalence implies Open Bisimilarity

We can show that

$$P \leq_\sigma Q \Longrightarrow P \preccurlyeq_\sigma Q$$

by showing that

$$\mathcal{S} \stackrel{\text{def}}{=} \biguplus_\sigma \{ (P, Q) \mid P \leq_\sigma Q \}$$

is a simulation. An immediate consequence is

$$\Gamma \vdash E \preccurlyeq^\circ E' :: \sigma \qquad \Longleftarrow \qquad \Gamma \vdash E \leq E' :: \sigma$$

Exercise: Think about result for open bisimilarity.

By **TB** and previous result,

$$P \rightsquigarrow P' \Longrightarrow P \approx_\sigma P' \Longrightarrow P \doteq_\sigma P' \quad \dagger$$

Hence if $P \leq_\sigma Q$ and $P \xrightarrow{\alpha} \xi$, then $P \rightsquigarrow^* V \xrightarrow{\alpha} \xi$, so $Q \rightsquigarrow^* V'$ for some V' using the empty context. Thus by $\dagger$,

$$V \doteq_\sigma P \quad \wedge \quad P \leq_\sigma Q \quad \wedge \quad Q \doteq_\sigma V$$

and so $V \leq_\sigma V'$. We then show that there is ξ'

$$V' \xrightarrow{\alpha} \xi' \quad \wedge \quad (\xi \leq_\sigma \xi' \vee \xi = \xi' = \top)$$

by a case analysis on V (and $Q \xrightarrow{\alpha} \xi$).

Case $V = P_1 : P_2$. Consider context

$$K \stackrel{\text{def}}{=} \mathsf{if\ elist}(v)\ \mathsf{then}\ \underline{0}\ \mathsf{else\ hd}(\mathsf{nil})$$

Use K to show any two contextually equivalent list expressions must either both be empty, or both be non-empty.

$$\begin{tikzcd}
P_1 : P_2 \arrow[d, "\mathsf{hd}"'] & \leq_{[\sigma]} & V' & = & Q_1 : Q_2 \arrow[d, "\mathsf{hd}"'] & \textit{From above} \\
P_1 & & & & Q_1 & \textit{Hence}
\end{tikzcd}$$

and by †, and definition of $\leq_{[\sigma]}$ with $C' \stackrel{\text{def}}{=} C[\mathsf{hd}(v)/v]$,

$$P_1 \doteq_\sigma \mathsf{hd}(P_1 : P_2) \leq_\sigma \mathsf{hd}(Q_1 : Q_2) \doteq_\sigma Q_1$$

Another Equivalence

$$\begin{aligned} \mathsf{N} &\equiv \underline{0} : \mathsf{M}\,\mathsf{S}\,\mathsf{N} \\ \mathsf{F}\,n &\equiv n : \mathsf{F}\,(n + \underline{1}) \\ \mathsf{S}\,n &\equiv n + \underline{1} \end{aligned}$$

Then

$$\mathsf{N} \doteq_{[\mathsf{int}]} \mathsf{F}\,\underline{0} \quad \Longleftarrow \quad \mathsf{N} \approx_{[\mathsf{int}]} \mathsf{F}\,\underline{0}$$

Define

$$\begin{aligned} \mathcal{B} \stackrel{\text{def}}{=} \{ \; & (P, Q),\ (\mathsf{F}\,\underline{0}, \mathsf{N}), & \mid P \approx Q \,\} \\ & (\mathsf{F}\,(\underline{0} + \underbrace{\underline{1} + \ldots + \underline{1}}_{i \geq 1}), \mathsf{M}\,\mathsf{S}\,(\underbrace{\mathsf{tl} \ldots \mathsf{tl}}_{i-1}(\mathsf{N}))) \end{aligned}$$

$$\mathsf{F}\,(\underline{0}+\underbrace{\underline{1}+\ldots+\underline{1}}_{i}) \rightsquigarrow \underline{0}+\underbrace{\underline{1}+\ldots+\underline{1}}_{i} : \mathsf{F}\,(\underline{0}+\underbrace{\underline{1}+\ldots+\underline{1}}_{i+1})$$

$$\xrightarrow{\mathsf{hd}} \underline{0}+\underbrace{\underline{1}+\ldots+\underline{1}}_{i}$$

$$\xrightarrow{\mathsf{tl}} \mathsf{F}\,(\underline{0}+\underbrace{\underline{1}+\ldots+\underline{1}}_{i+1})$$

$$\mathsf{M\,S}\,(\underbrace{\mathsf{tl}\ldots\mathsf{tl}}_{i-1}(\mathsf{N})) \rightsquigarrow \mathsf{S}\,(\mathsf{hd}(\underbrace{\mathsf{tl}\ldots\mathsf{tl}}_{i-1}(\mathsf{N}))) : \mathsf{M\,S}\,(\underbrace{\mathsf{tl}\ldots\mathsf{tl}}_{i}(\mathsf{N}))$$

$$\xrightarrow{\mathsf{hd}} \mathsf{S}\,(\mathsf{hd}(\underbrace{\mathsf{tl}\ldots\mathsf{tl}}_{i-1}(\mathsf{N})))$$

$$\xrightarrow{\mathsf{tl}} \mathsf{M\,S}\,(\underbrace{\mathsf{tl}\ldots\mathsf{tl}}_{i}(\mathsf{N}))$$

Exercise: Show heads bisimilar.

Overview Part III

- Here is a brief survey of a few references . . .

[GA97] A calculus for cryptographic protocols: The spi calculus.

- Suppose $\{D\}_K$ is data encrypted with a key K (ciphertext).
- Suppose that $(\nu c)P$ is any process P with private channel c.
- The process $\overline{c}\langle\{D\}_K\rangle$ outputs $\{D\}_K$ on c. Then ...

$$\overline{c}\langle\{D\}_K\rangle \sim \overline{c}\langle\{D'\}_K\rangle$$

- Paper describes the spi calculus ...
- Secrecy properties are captured by process equivalences. Restricted channels do not reveal data:

$$(\nu c)(\overline{c}\langle M\rangle \mid c(x).F(x)) \sim (\nu c)(\overline{c}\langle M'\rangle \mid c(x).F(x)) \Longleftarrow F(M) \sim F(M')$$

[GA98] A Bisimulation Method for Cryptographic Protocols

- Results based around the spi calculus.
- Refines "our notion" of bisimulation: matching actions replaced by indistinguishable actions (privacy).
- Bisimulations relative to a set of names;
- and relations specifying that environments cannot distinguish certain (encrypted) data.
- Gives examples of bisimulation equivalences.

[GC00] Mobile Ambients

- Defines the ambient calculus . . .
- an ambient $n[P]$ is a bounded "process"; security is represented by the possibilities of crossing boundaries.
- Again, contextual equivalence is a key notion . . .

[GC03] Equational Properties of Mobile Ambients

- Reviews the ambient calculus.
- Develops a theory for reasoning about contextual equivalence, and gives some examples.

[RTJ01] The Coalgebraic Class Specification Language CCSL

- Introduces Coalgebraic Class Specification Language (CCSL).
- Allows the user to "specify coalgebras" . . .
- and associated bisimulations.
- The specifications are compiled into PVS or Isabelle.
- CCSL has been used to verify security properties.

[Gim95] Coinductive Types in Coq: An Experiment with the Alternating Bit Protocol

- Develops a proof of the Alternating Bit Protocol within Coq.

[Cro99] Operational Semantics

- A basic introduction to Plotkin style operational semantics.
- Lot's of detail, with an easy pace.
- Includes imperative languages.

[Cro98] Lectures on [Co]Induction and [Co]Algebras

- Basic operational semantics via (co)inductive definitions.
- Defines, with examples, algebras and coalgebras.
- Briefly outlines categorical induction and coinduction.

[Gor95] Bisimilarity as a theory of functional programming

- Tutorial on labelled transition semantics:detailed.
- Similar in flavour to these lectures, but …
- covers theory to a greater depth.
- Many examples of equivalences via coinduction.

[Pit97] Operationally Based Theories of Program Equivalence

- Tutorial, with bisimilarity founded on “evaluation”.
- Two expressions are bisimilar if they evaluate to values, and all “subexpressions” bisimilar.
- Explains “continuity” properties of $\mathsf{fix}.E$ by syntactic methods.

[CG95, CG99] Relating Operational and Denotational Semantics for Input/Output Effects

- Original conference and journal versions of ideas outlined here.
- Covers labelled transition semantics …
- for a functional language with imperative I/O.
- Also includes a denotational model and adequacy results.

[Cro01] Completeness of Bisimilarity for Contextual Equivalence in Linear Theories

- Similar, for a linear language, with bisimilarity based on evaluation.

References

[CG95] R. L. Crole and A. D. Gordon. A Sound Metalogical Semantics for Input/Output Effects. In L. Pacholski and J. Tiuryn, editors, *Proceedings of Computer Science Logic 1994*, volume 933 of *Lecture Notes in Computer Science*, pages 339–353. Springer-Verlag, 1995.

[CG99] R. L. Crole and A. D. Gordon. Relating Operational and Denotational Semantics for Input/Output Effects. *Mathematical Structures in Computer Science*, 9:125–158, 1999.

[Cro98] R. L. Crole. Lectures on [Co]Induction and [Co]Algebras. Technical Report 1998/12, Department of Mathematics and Computer Science, University of Leicester, 1998.

[Cro99] R. L. Crole. Operational Semantics, 1999. Department of Mathematics and Computer Science Lecture Notes, LaTeX format 101 pages with subject and notation index.

[Cro01] R. L. Crole. Completeness of Bisimilarity for Contextual Equivalence in Linear Theories. *Electronic Journal of the IGPL*, 9(1), January 2001.

[GA97] A. D. Gordon and M. Abadi. A calculus for cryptographic protocols: The spi calculus. In *Proceedings of the Fourth ACM Conference on Computer and Communications Security*, pages 36–47. ACM Press, 1997.

[GA98] A. D. Gordon and M. Abadi. A Bisimulation Method for Cryptographic Protocols. *Nordic Journal of Computing*, 5:267–303, 1998. Original version in Proceedings ESOP'98, Springer LNCS.

[GC00] A. D. Gordon and L. Cardelli. Mobile Ambients. *Theoretical Computer Science*, 240:177–213, 2000.

[GC03] A. D. Gordon and L. Cardelli. Equational Properties of Mobile Ambients. *Mathematical Structures in Computer Science*, 13(3):371–408, 2003.

[Gim95] E. Giménez. Coinductive Types in Coq: An Experiment with the Alternating Bit Protocol. Ecole Normale Supérieure de Lyon, Research Report No. 95-38, 1995.

[Gor95] A. D. Gordon. Bisimilarity as a theory of functional programming. Technical report, Aarhus University, Denmark, 1995. BRICS Notes Series NS–95–3, BRICS, Aarhus University.

[Pit97] A. M. Pitts. Operationally Based Theories of Program Equivalence. In P. Dybjer and A. M. Pitts, editors, *Semantics and Logics of Computation*, 1997.

[RTJ01] J. Rothe, H. Tews, and B. Jacobs. The Coalgebraic Class Specification Language CCSL. *Journal of Universal Computer Science*, 7(2):175–193, February 2001.