Recitation #14:
Concurrent Algol

15-312: Principles of Programming Languages

Wednesday, April 24th, 2013

1 Syntax and Semantics

There are two new types of interest to us in Concurrent Algol: channels and events. *Channels* represent a means of communication. Associated with a channel is a type τ, representing the type of data that may be send over a given channel. We generalize the notion of receiving a message on a channel to waiting for an *event* to allow us to do things like respond to a message received on any of a list of channels (all of which must carry data of the same type) and to deal with channel references (so that you don't have to statically specify which channel you're using, thereby allowing dynamic communication graphs).

Expressions can be an encapsulated command ($\mathsf{cmd}(m)$), a reference to a channel, ($\mathsf{chan}[a]$, concrete syntax $\&a$) or an event, consisting of a choice made up of a request to receive a message along a channel, ($\mathsf{rcvref}(e)$, concrete syntax $??e$), or recieve a message along a channel ($\mathsf{sendref}(e; e')$, concrete syntax $!!e(e')$). The expression $\mathsf{or}(e_1; e_2)$ represents a binary choice, and $\mathsf{never}[\tau]$ represents the nullary choice.

The core of the modal separation is the monadic structure consisting of the return command $\mathsf{ret}(e)$ and the sequencing command $\mathsf{bnd}(e; x.m)$. Modernized Algol added memory manipulation commands. Concurrent Algol instead adds the abiliy to create a channel $\mathsf{newchn}[\tau]$, to wait for an event ($\mathsf{sync}(e)$) and to spawn a new process ($\mathsf{spawn}(e)$, concrete syntax $\mathbf{0}$).

A process can either be idle (stop, concrete syntax $\mathbf{1}$), consist of a single command ($\mathsf{proc}(m)$), or consist of two processes running concurrently ($p_1 \parallel p_2$). Channels are declared at the process-level by the form ($\nu\ \tau \sim a\{p\}$).

1.1 Statics

The statics of the language are defined with three judgements. As usual, $\Gamma \vdash_\Sigma e : \tau$ is the typing rule for expressions. The signature $\Sigma = a_1 \sim \tau_1, ..., a_n \sim \tau_n$ describes the names and associated types of channels that are available for use. The rules for the new forms are given below.

$\boxed{\Gamma \vdash_\Sigma e : \tau}$

$$\frac{\Gamma \vdash_\Sigma m \sim \tau}{\Gamma \vdash_\Sigma \mathsf{cmd}(m) : \mathsf{cmd}(\tau)}(\mathrm{S}_1) \qquad \frac{}{\Gamma \vdash_{\Sigma, a\sim\tau} \mathsf{chan}[a] : \mathsf{chan}(\tau)}(\mathrm{S}_2) \qquad \frac{\Gamma \vdash_\Sigma e : \mathsf{chan}(\tau)}{\Gamma \vdash_\Sigma \mathsf{rcvref}(e) : \mathsf{event}(\tau)}(\mathrm{S}_3)$$

$$\frac{\Gamma \vdash_\Sigma e : \mathsf{chan}(\tau) \qquad \Gamma \vdash_\Sigma e' : \tau}{\Gamma \vdash_\Sigma \mathsf{sendref}(e; e') : \mathsf{event}(\tau)}(\mathrm{S}_4) \qquad \frac{\Gamma \vdash_\Sigma e_1 : \mathsf{event}(\tau) \qquad \Gamma \vdash_\Sigma e_2 : \mathsf{event}(\tau)}{\Gamma \vdash_\Sigma \mathsf{or}(e_1; e_2) : \mathsf{event}(\tau)}(\mathrm{S}_5)$$

The judgement $\Gamma \vdash_\Sigma m \sim \tau$ is the monadic typing of a command.

Sort			**Abstract Syntax**	**Description**
Type	τ	::=	...	(PCF types)
			$\mathsf{cmd}(\tau)$	command
			$\mathsf{event}(\tau)$	event
			$\mathsf{chan}(\tau)$	channel
Exp	e	::=	...	(PCF terms)
			$\mathsf{cmd}(m)$	encapsulation
			$\mathsf{chan}[a]$	channel reference
			$\mathsf{sendref}(e_1; e_2)$	send event
			$\mathsf{rcvref}(e)$	receive event
			$\mathsf{never}[\tau]$	nullary disjunction event
			$\mathsf{or}(e_1; e_2)$	binary disjuction event
Cmd	m	::=	$\mathsf{ret}(e)$	return
			$\mathsf{bnd}(e; x.m)$	sequence
			$\mathsf{spawn}(e)$	spawn process
			$\mathsf{sync}(e)$	wait for event
			$\mathsf{newchn}[\tau]$	new channel
Proc	p	::=	stop	idle
			$\mathsf{proc}(m)$	atomic
			$p_1 \mid\mid p_2$	parallel
			$\nu\, a \sim \tau\{p\}$	new channel
Action	α	::=	ϵ	none
			$a!e$	send
			$a?e$	receive

$\boxed{\Gamma \vdash_\Sigma m \sim \tau}$

$$\frac{\Gamma \vdash_\Sigma e : \tau}{\Gamma \vdash_\Sigma \mathsf{ret}(e) \sim \tau}(\mathrm{S}_6) \qquad \frac{\Gamma \vdash_\Sigma e : \mathsf{cmd}(\tau) \qquad \Gamma, x : \tau \vdash_\Sigma m \sim \tau'}{\Gamma \vdash_\Sigma \mathsf{bnd}(e; x.m) \sim \tau'}(\mathrm{S}_7) \qquad \frac{\Gamma \vdash_\Sigma e : \mathsf{cmd}(\mathsf{unit})}{\Gamma \vdash_\Sigma \mathsf{spawn}(e) \sim \mathsf{unit}}(\mathrm{S}_8)$$

$$\frac{\Gamma \vdash_\Sigma e : \mathsf{event}(\tau)}{\Gamma \vdash_\Sigma \mathsf{sync}(e) \sim \tau}(\mathrm{S}_9) \qquad \frac{}{\Gamma \vdash_\Sigma \mathsf{newchn}[\tau] \sim \mathsf{chan}(\tau)}(\mathrm{S}_{10})$$

The new judgement $p\ \mathsf{proc}_\Sigma$ describes the valid processes relative to the signature Σ.

$\boxed{p\ \mathsf{proc}_\Sigma}$

$$\frac{}{\mathsf{stop}\ \mathsf{proc}_\Sigma}(\mathrm{S}_{11}) \qquad \frac{\emptyset \vdash_\Sigma m \sim \tau}{\mathsf{proc}(m)\ \mathsf{proc}_\Sigma}(\mathrm{S}_{12}) \qquad \frac{p_1\ \mathsf{proc}_\Sigma \qquad p_2\ \mathsf{proc}_\Sigma}{p_1 \mid\mid p_2\ \mathsf{proc}_\Sigma}(\mathrm{S}_{13}) \qquad \frac{p\ \mathsf{proc}_{\Sigma, a \sim \tau}}{\nu\, a \sim \tau\{p\}\ \mathsf{proc}_\Sigma}(\mathrm{S}_{14})$$

Processes are identified up to structural congruence, an equivalence relation written $p_1 \equiv p_2$. It includes the familiar notion of α-equivalence (E_4) and adds rules to discard any processes that have completed (E_5), conduct scope extrusion for channels (E_7), and discard channels that can no longer be used (E_8).

$\boxed{p_1 \equiv p_2}$

$$\frac{}{p_1 \equiv p_1}(\mathrm{E}_1) \qquad \frac{p_2 \equiv p_1}{p_1 \equiv p_2}(\mathrm{E}_2) \qquad \frac{p_1 \equiv p_2 \qquad p_2 \equiv p_3}{p_1 \equiv p_3}(\mathrm{E}_3) \qquad \frac{p_1 =_\alpha p_2}{p_1 \equiv p_2}(\mathrm{E}_4) \qquad \frac{}{p \mid\mid \mathsf{stop} \equiv p}(\mathrm{E}_5)$$

$$\frac{p_1 \equiv p_1' \qquad p_2 \equiv p_2'}{p_1 \mid\mid p_2 \equiv p_1' \mid\mid p_2'}(\mathrm{E}_6) \qquad \frac{a \notin p_2}{\nu\ a \sim \tau\{p_1\} \mid\mid p_2 \equiv \nu\ a \sim \tau\{(p_1 \mid\mid p_2)\}}(\mathrm{E}_7) \qquad \frac{a \notin p}{\nu\ a \sim \tau\{p\} \equiv p}(\mathrm{E}_8)$$

Actions, used in the dynamic below, also admit a statics, given by the judgement $\vdash_\Sigma \alpha\ \mathsf{action}$.

$\boxed{\vdash_\Sigma \alpha\ \mathsf{action}}$

$$\frac{}{\vdash_\Sigma \epsilon\ \mathsf{action}}(\mathrm{E}_9) \qquad \frac{\emptyset \vdash_{\Sigma, a \sim \tau} e : \tau}{\vdash_{\Sigma, a \sim \tau} a!e\ \mathsf{action}}(\mathrm{E}_{10}) \qquad \frac{\emptyset \vdash_{\Sigma, a \sim \tau} e : \tau}{\vdash_{\Sigma, a \sim \tau} a?e\ \mathsf{action}}(\mathrm{E}_{11})$$

1.2 Dynamics

The dynamics of Concurrent Algol are governed by five judgements. The value judgement $e\ \mathsf{val}_\Sigma$, as usual, describes the conditions in which the expression e is a value under channel signature, Σ.

$\boxed{e\ \mathsf{val}_\Sigma}$

$$\frac{}{\mathsf{chan}[a]\ \mathsf{val}_{\Sigma, a \sim \tau}}(\mathrm{V}_1) \qquad \frac{e\ \mathsf{val}_\Sigma}{\mathsf{rcvref}(e)\ \mathsf{val}_\Sigma}(\mathrm{V}_2) \qquad \frac{e\ \mathsf{val}_\Sigma \qquad e'\ \mathsf{val}_\Sigma}{\mathsf{sendref}(e; e')\ \mathsf{val}_\Sigma}(\mathrm{V}_3) \qquad \frac{}{\mathsf{never}[\tau]\ \mathsf{val}_\Sigma}(\mathrm{V}_4)$$

$$\frac{e_1\ \mathsf{val}_\Sigma \qquad e_2\ \mathsf{val}_\Sigma}{\mathsf{or}(e_1; e_2)\ \mathsf{val}_\Sigma}(\mathrm{V}_5) \qquad \frac{}{\mathsf{cmd}(m)\ \mathsf{val}_\Sigma}(\mathrm{V}_6)$$

The other four judgements are transition systems that describe how expressions, commands, or processes may take a step. An expression may take an (effect-free) step by the judgement $e \underset{\Sigma}{\mapsto} e'$.

$\boxed{e \underset{\Sigma}{\mapsto} e'}$

$$\frac{e \underset{\Sigma}{\mapsto} e'}{\mathsf{rcvref}(e) \underset{\Sigma}{\mapsto} \mathsf{rcvref}(e')}(\mathrm{D}_1) \qquad \frac{e_1 \underset{\Sigma}{\mapsto} e_1'}{\mathsf{sendref}(e_1; e_2) \underset{\Sigma}{\mapsto} \mathsf{sendref}(e_1'; e_2)}(\mathrm{D}_2)$$

$$\frac{e_1\ \mathsf{val}_\Sigma \qquad e_2 \underset{\Sigma}{\mapsto} e_2'}{\mathsf{sendref}(e_1; e_2) \underset{\Sigma}{\mapsto} \mathsf{sendref}(e_1'; e_2)}(\mathrm{D}_3) \qquad \frac{e_1 \underset{\Sigma}{\mapsto} e_1'}{\mathsf{or}(e_1; e_2) \underset{\Sigma}{\mapsto} \mathsf{or}(e_1'; e_2)}(\mathrm{D}_4) \qquad \frac{e_1\ \mathsf{val}_\Sigma \qquad e_2 \underset{\Sigma}{\mapsto} e_2'}{\mathsf{or}(e_1; e_2) \underset{\Sigma}{\mapsto} \mathsf{or}(e_1; e_2')}(\mathrm{D}_5)$$

The judgement $m \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} m'$ describes the evaluation of a command m under signature Σ. Commands are capable of actions (which will engender an effect), given by α.

$\boxed{m \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} m'}$

$$\frac{e \underset{\Sigma}{\mapsto} e'}{\mathsf{ret}(e) \overset{\epsilon}{\underset{\Sigma}{\Rightarrow}} \mathsf{ret}(e')}(\mathrm{M}_1) \qquad \frac{e \underset{\Sigma}{\mapsto} e'}{\mathsf{bnd}(e; x.m) \overset{\epsilon}{\underset{\Sigma}{\Rightarrow}} \mathsf{bnd}(e'; x.m)}(\mathrm{M}_2) \qquad \frac{e\ \mathsf{val}_\Sigma}{\mathsf{bnd}(\mathsf{cmd}(\mathsf{ret}(e)); x.m) \overset{\epsilon}{\underset{\Sigma}{\Rightarrow}} [e/x]m}(\mathrm{M}_3)$$

$$\frac{m_1 \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} m_1'}{\mathsf{bnd}(\mathsf{cmd}(m_1); x.m_2) \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} \mathsf{bnd}(\mathsf{cmd}(m_1'); x.m_2)}(\mathrm{M}_4) \qquad \frac{e \underset{\Sigma}{\mapsto} e'}{\mathsf{spawn}(e) \overset{\epsilon}{\underset{\Sigma}{\Rightarrow}} \mathsf{spawn}(e')}(\mathrm{M}_5)$$

$$\frac{e\ \mathsf{val}_\Sigma \qquad e \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} m}{\mathsf{sync}(e) \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} m}(\mathrm{M}_6)$$

In the final rule above, the possible actions engendered by an event value are defined by the auxiliary judgement $e \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} m$, which states that the event value e engenders action α and activates command m.

$\boxed{e \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} m}$

$$\frac{}{\mathsf{rcvref}(\mathsf{chan}[a]) \overset{a?e}{\underset{\Sigma, a\sim\tau}{\Rightarrow}} \mathsf{ret}(e)}(\mathrm{M}_7) \qquad \frac{e\ \mathsf{val}_{\Sigma, a\sim\tau}}{\mathsf{sendref}(\mathsf{chan}[a]; e) \overset{a!e}{\underset{\Sigma, a\sim\tau}{\Rightarrow}} \mathsf{ret}(e)}(\mathrm{M}_8)$$

$$\frac{e_1 \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} m}{\mathsf{or}(e_1; e_2) \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} m}(\mathrm{M}_9) \qquad \frac{e_2 \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} m}{\mathsf{or}(e_1; e_2) \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} m}(\mathrm{M}_{10})$$

Let us now consider the dynamics of processes themselves. The judgement $m \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} \nu\ \Sigma'\{m' \mid\mid p\}$ says that the command m may spawn new processes p and introduce new channels given by Σ'.

$\boxed{m \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} \nu\ \Sigma'\{m' \mid\mid p\}}$

$$\frac{m_1 \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} \nu\ \Sigma'\{m_1' \mid\mid p\}}{\mathsf{bnd}(\mathsf{cmd}(m_1); x.m_2) \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} \nu\ \Sigma'\{\mathsf{bnd}(\mathsf{cmd}(m_1'); x.m_2) \mid\mid p\}}(\mathrm{M}_{11})$$

$$\frac{}{\mathsf{spawn}(\mathsf{cmd}(m)) \overset{\epsilon}{\underset{\Sigma}{\Rightarrow}} \nu\ \cdot \{\mathsf{ret}(\langle\rangle) \mid\mid \mathsf{proc}(m)\}}(\mathrm{M}_{12}) \qquad \frac{}{\mathsf{newchn}[\tau] \overset{\epsilon}{\underset{\Sigma}{\Rightarrow}} \nu\ a \sim \tau\{\mathsf{ret}(\mathsf{chan}[a]) \mid\mid \mathsf{stop}\}}(\mathrm{M}_{13})$$

This auxiliary judgement allows us to define the stepping rules for processes themselves, given by the judgement $p \overset{\alpha}{\underset{\Sigma}{\mapsto}} p'$. The notation $\nu\ \Sigma\{p\}$ is shorthand for the repeated iteration of $\nu\ a \sim \tau\{p\}$. That is, it defined as $\nu\ a_1 \sim \tau_1\{...\nu\ a_n \sim \tau_n\{p\}\}$ when $\Sigma = a_1 \sim \tau_1, ..., a_n \sim \tau_n$. When Σ is empty, it is simply p.

$\boxed{p \overset{\alpha}{\underset{\Sigma}{\mapsto}} p'}$

$$\frac{m \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} m'}{\mathsf{proc}(m) \overset{\alpha}{\underset{\Sigma}{\mapsto}} \mathsf{proc}(m')}(\mathrm{P}_1) \qquad \frac{m \overset{\alpha}{\underset{\Sigma}{\Rightarrow}} \nu\, \Sigma'\{m' \mid\mid p\}}{\mathsf{proc}(m) \overset{\alpha}{\underset{\Sigma}{\mapsto}} \nu\, \Sigma'\{\mathsf{proc}(m') \mid\mid p\}}(\mathrm{P}_2) \qquad \frac{e\ \mathsf{val}_\Sigma}{\mathsf{proc}(\mathsf{ret}(e)) \overset{\epsilon}{\underset{\Sigma}{\mapsto}} \mathsf{stop}}(\mathrm{P}_3)$$

$$\frac{p_1 \overset{\alpha}{\underset{\Sigma}{\mapsto}} p_1'}{p_1 \mid\mid p_2 \overset{\alpha}{\underset{\Sigma}{\mapsto}} p_1' \mid\mid p_2}(\mathrm{P}_4) \qquad \frac{p_1 \overset{a!e}{\underset{\Sigma}{\mapsto}} p_1' \qquad p_2 \overset{a?e}{\underset{\Sigma}{\mapsto}} p_2'}{p_1 \mid\mid p_2 \overset{\epsilon}{\underset{\Sigma}{\mapsto}} p_1' \mid\mid p_2'}(\mathrm{P}_5) \qquad \frac{p \overset{\alpha}{\underset{\Sigma, a\sim\tau}{\mapsto}} p' \qquad \vdash_\Sigma \alpha\ \mathsf{action}}{\nu\, a \sim \tau\{p\} \overset{\alpha}{\underset{\Sigma}{\mapsto}} \nu\, a \sim \tau\{p'\}}(\mathrm{P}_6)$$