CASheW-s

The University Of Sheffield.

Composition and Semantic Enhancement of Web-Services

THE UNIVERSITY OF SHEFFIELD

Department of

computer science

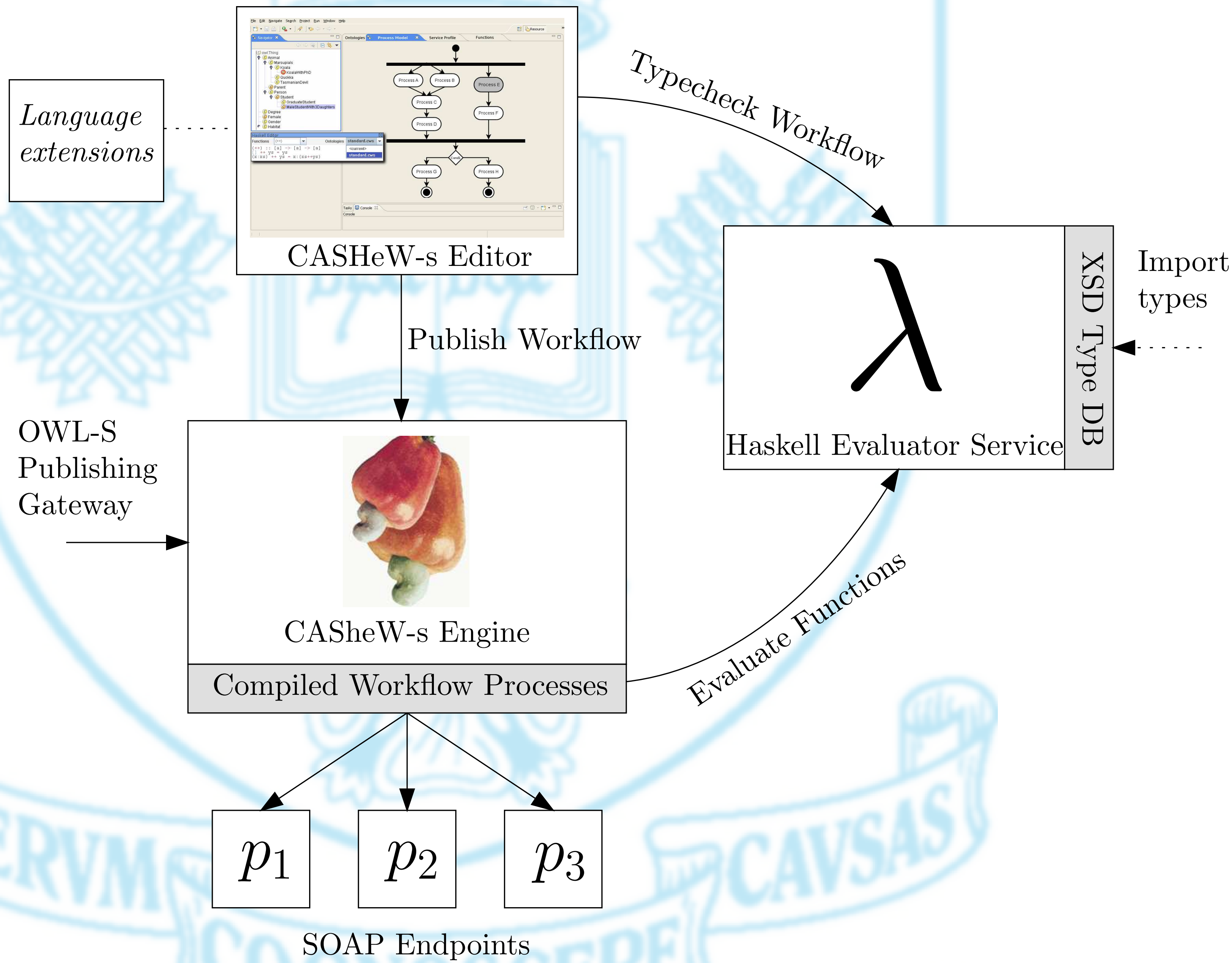
Introduction to the CASheW-s Project

- Our main objective is to develop a more generic approach to Web-Service composition.
- Therefore we are investigating the use of a timed process calculus to provide *compositional* behavioural semantics for workflows.
- The culmination of this will be a workflow engine,which will first be able to orchestrate OWL-S workflows.
- In this presentation we look at the operational semantics for OWL-S, and our approach to building them.

CaSHew-NUtS

- A conservative extension of the timed process calculus CaSE, which itself is a conservative extension of Milner's CCS.
- Extends CCS with the notion of abstract clocks, which facilitate multi-party synchronization.
- In CaSE, clocks are bound by *maximal progress*, meaning silent actions always take precedence over clock ticks.
- CaSHew-NUtS extends this concept with the possibility of clocks which do not exhibit maximal progress.

CASheW-s Architecture



CaSHew-NUtS Composition Rules

$$\mathsf{Com3}\ \frac{E \xrightarrow{a} E', F \xrightarrow{\overline{a}} F'}{E \mid F \xrightarrow{\tau} E' \mid F'} \qquad \begin{aligned} \Lambda &= \{a, b, c, \cdots\} \\ \overline{\Lambda} &= \{\overline{a}, \overline{b}, \overline{c}, \cdots\} \end{aligned}$$

$$\alpha \in \mathcal{A} = \Lambda \cup \overline{\Lambda} \cup \{\tau\}$$

$$\mathsf{Com1}\ \frac{E \xrightarrow{\alpha} E'}{E \mid F \xrightarrow{\alpha} E' \mid F} \qquad \mathsf{Com2}\ \frac{F \xrightarrow{\alpha} F'}{E \mid F \xrightarrow{\alpha} E \mid F'}$$

$$\mathsf{Com4}\ \frac{E \xrightarrow{\sigma_i} E' \quad F \xrightarrow{\sigma_j} F'}{E \mid F \xrightarrow{\sigma_{i \cdot j \cdot k}} E' \mid F'}$$

$$\gamma \in \mathcal{A} \cup \mathcal{T}$$

$$\mathcal{T} = \{\rho, \sigma, \cdots\}$$

CASheW-s Syntax

- Problems with OWL-S Syntax
  - Incoming dataflow tied to Performance restricting further composition.
  - Fine for persistence/communication, but doesn't represent the composition of a system.
  - Uncomfortable notion of *Produce* tied to dummy variable *TheParentPerform*.
- CASheW-s syntax
  - More open to composition.
  - Allows compositional translation from OWL-S syntax.

Process Syntax for CASheW-s

$$
\begin{array}{rcl}
Process & ::= & \textbf{AtomicProcess}\ m\ AProcess \mid \\
 & & \textbf{CompositeProcess}\ m\ CProcess \\
 & & \quad ConsumeList\ ProduceList \\
CProcess & ::= & \textbf{Sequence}\ PerformanceList \mid \\
 & & \textbf{Split}\ PerformanceList \mid \\
 & & \textbf{SplitJoin}\ PerformanceList \mid \\
 & & \textbf{Any-Order}\ PerformanceList \mid \\
 & & \textbf{ChooseOne}\ PerformanceList \mid \\
 & & \textbf{IfThenElse}\ Performance\ Performance \mid \\
 & & \textbf{RepeatWhile}\ Performance \mid \\
 & & \textbf{RepeatUntil}\ Performance
\end{array}
$$

Performance Syntax for CASheW-s

$$
\begin{array}{rcl}
Performance & ::= & \textbf{Perform}\ n\ Process\ DataAggregation \\
Connection & ::= & \textbf{Connect}\ n\ c\ o\ a\ j \\
PerformanceList & ::= & Performance \mid \\
 & & (PerformanceList); Performance \mid \\
 & & (PerformanceList); Connection \\
DataAggregation & ::= & ValueDataList \\
 & & ValueCollectorList \\
ValueData & ::= & \textbf{ValueData}\ a \\
ValueDataList & ::= & \epsilon \mid ValueData\ ValueDataTail \\
ValueDataTail & ::= & \epsilon \mid ;\ ValueData\ ValueDataTail \\
ValueCollector & ::= & \textbf{ValueCollector}\ a\ k \\
ValueCollectorList & ::= & \epsilon \mid ValueCollector\ ValueCollectorTail \\
ValueCollectorTail & ::= & \epsilon \mid ;\ ValueCollector\ ValueCollectorTail \\
Consume & ::= & \textbf{Consume}\ a\ n\ b\ j \\
ConsumeList & ::= & \epsilon \mid Consume\ ConsumeTail \\
ConsumeTail & ::= & \epsilon \mid ;\ Consume\ ConsumeTail \\
Produce & ::= & \textbf{Produce}\ c\ n\ d \\
ProduceList & ::= & \epsilon \mid Produce\ ProduceTail \\
ProduceTail & ::= & \epsilon \mid ;\ Produce\ ProduceTail
\end{array}
$$

Orchestration Channels

- **r** is the *ready to execute* channel, which a process uses to indicate that it has no further execution pre-conditions. (Something the informal semantics rely on, but no-one else has formalised).
- **e** is the *permission to execute* channel, which a process must receive input on before it can begin executing.
- **t** signifies the *token*, which signifies permission to execute for each of the process's child performances (in a similar fashion to a token ring network). Different token passing games facilitate performance serialization.

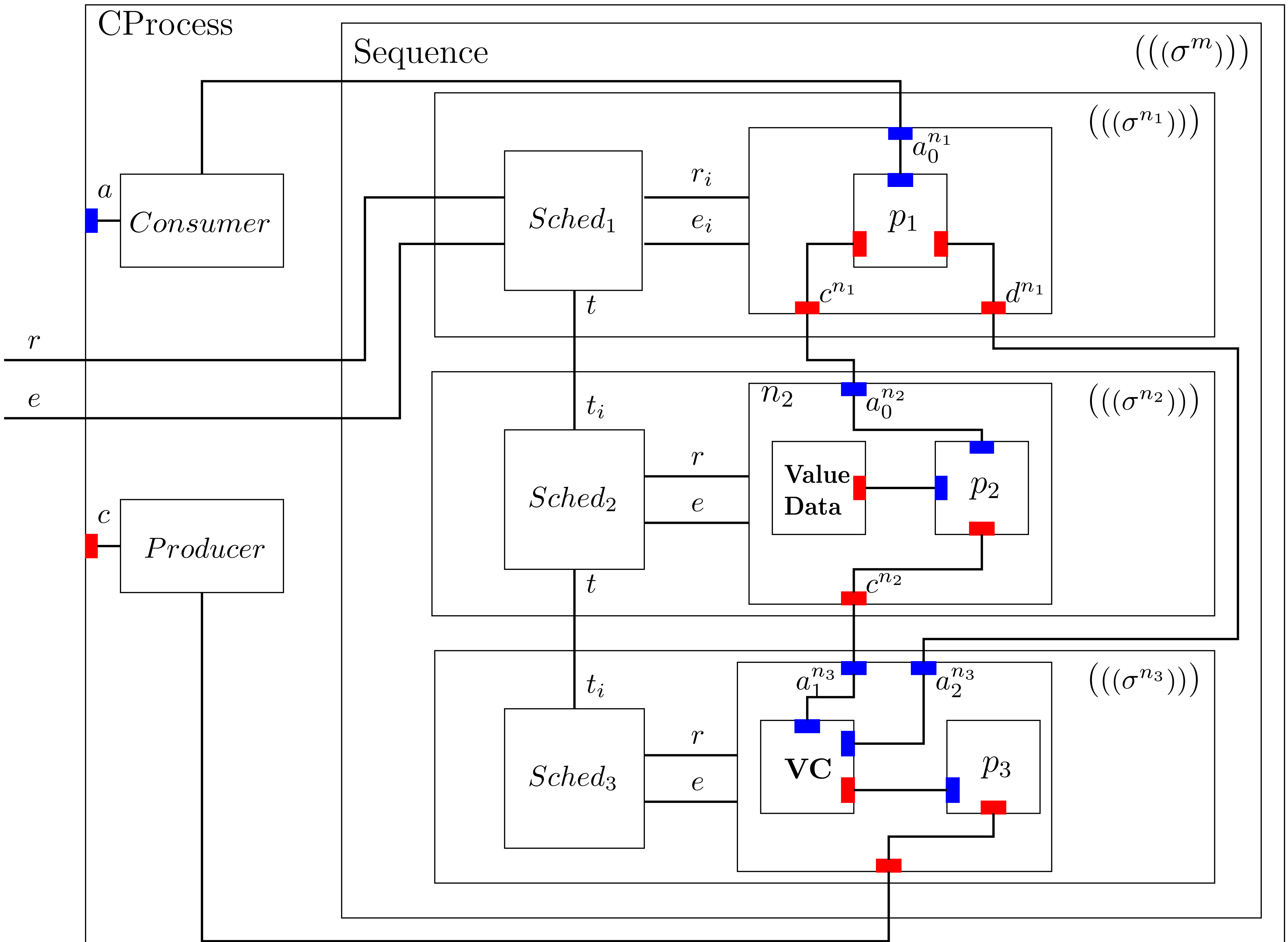
Orchestration Clocks

- Two main clock types used for orchestration.
- σ^m is the *process clock*, it ranges over the entire scope of the process and its child performances and may be used to resynchronize (such as after a split-join), where m is the name of the process.
- σ^n, σ^o are *performance clocks*, they are used to signal that a performance has completed, and are used to decide when control can be passed on to another scheduler in the system, where n and o are names of performances.

Composite Process Layout

CompositeProcess cp
    Sequence
        Perform n_1 p_1
        Connect c n_1 a n_2 0
        Connect d n_1 a n_3 2
        Perform n_2 p_2
            ValueData a
        Connect c n_2 a n_3 1
        Perform n_3 p_2
            ValueCollector a 2
    Consume a n_1 a 0
    Produce c n_3 c

Composite Process Layout in CASheW-s



OWL-S Process Semantics

$$[\![\mathbf{AtomicProcess\ m\ P}]\!]_C^A = {}^m[\![\mathbf{P}]\!]_C^A$$

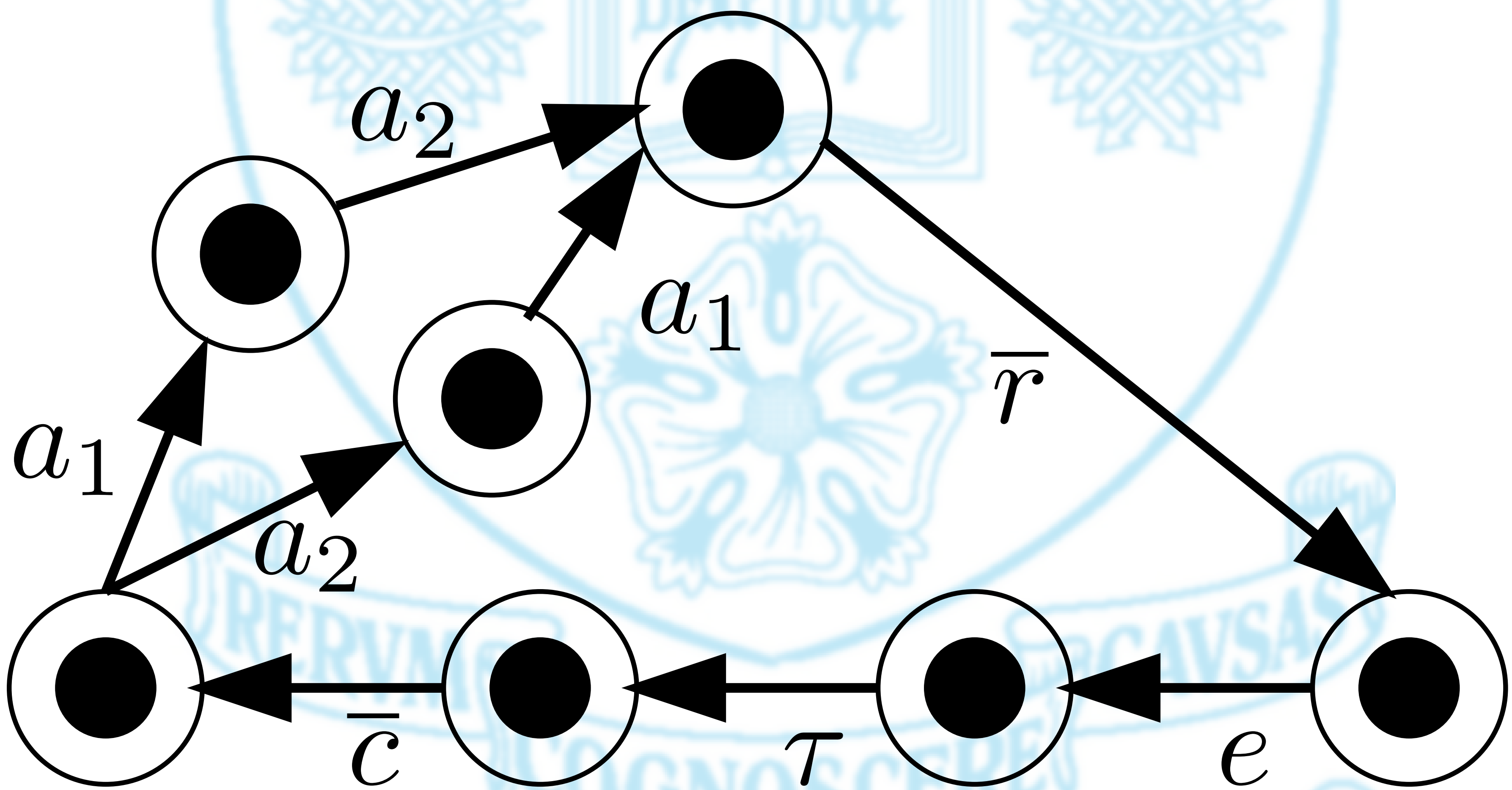
$$[\![\mathbf{CompositeProcess\ m\ P\ G\ H}]\!]_C^A = \\ ({}^m[\![\mathbf{P}]\!]_{C^m}^{A^m} \mid [\![\mathbf{G}]\!]_\emptyset^A \mid [\![\mathbf{H}]\!]_C^\emptyset) \setminus A^m \cup C^m / \{\sigma^c \mid c \in C\}$$

Where

- m is a process name
- p is a process
- A is a set of inputs
- C is a set of outputs
- G is a *Consume List*
- H is a *Produce List*

Example Atomic Process Semantics

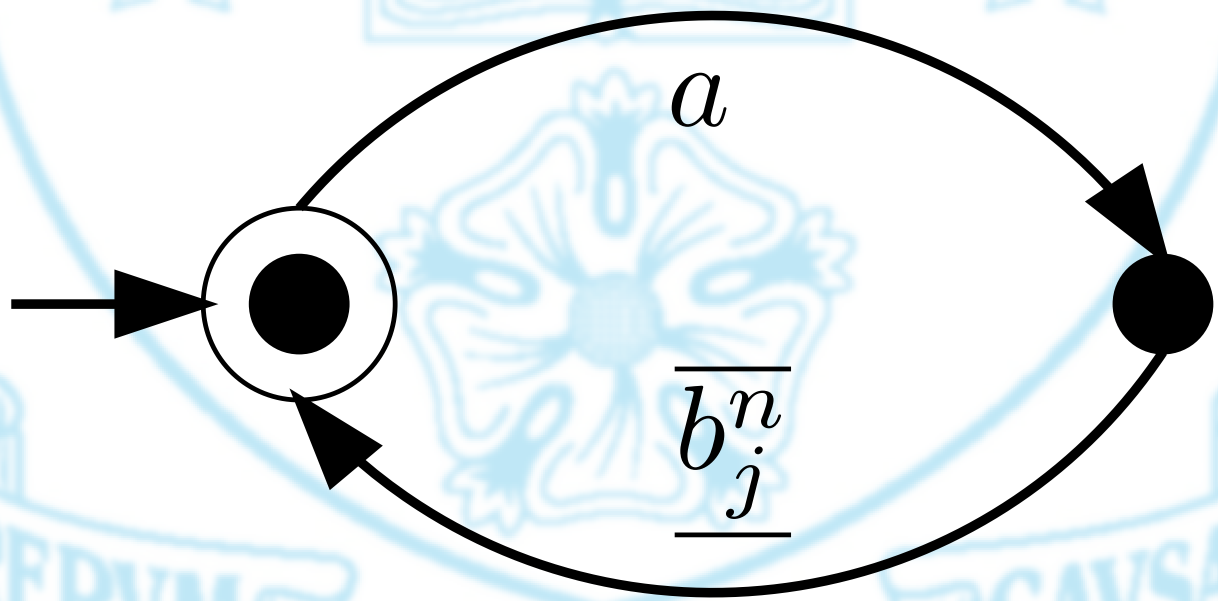
$$[\![AnAtomicProcess]\!]^{\{a_1,a_2\}}_{\{c\}} = \mu X. < a_1, a_2 > .\overline{r}.e.\tau.\overline{c}.X$$



Consume Semantics

- *Consume* pulls an input which is required to run a process.

$$[\![\mathbf{Consume}\ a\ n\ b\ j]\!]_{\emptyset}^{\{a\}} = \mu X.a.\underline{\overline{b_j^n}}.X$$

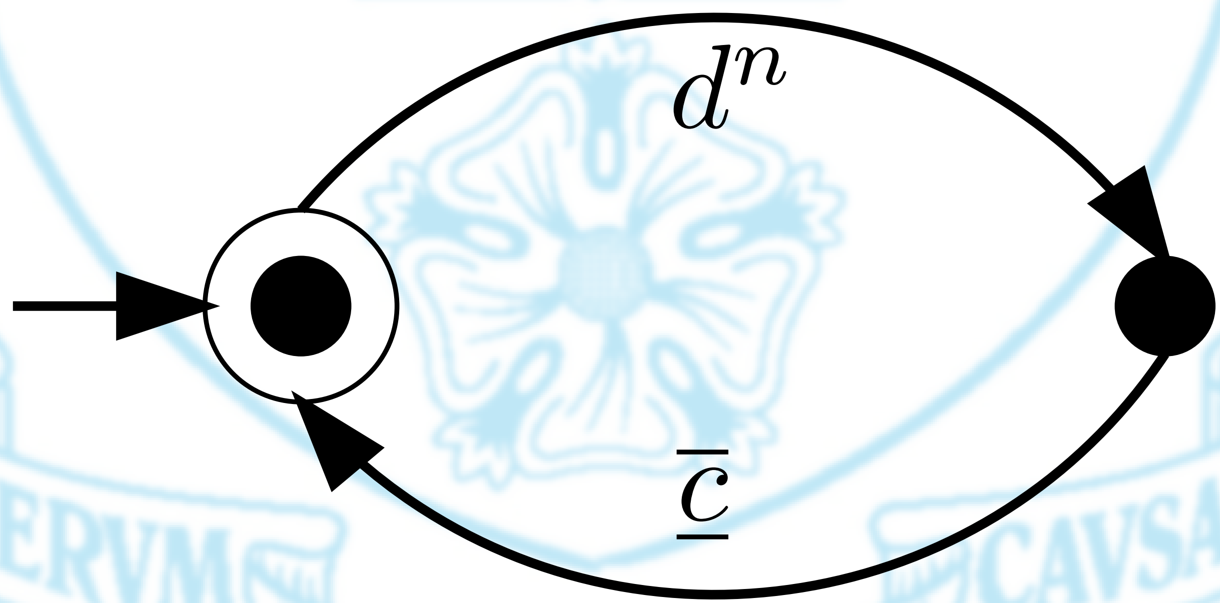


- Wires like Consume, patiently wait for input and then insistently output.

Produce Semantics

- *Produce* pushes an output which has been produced by a process.

$$[\![\mathbf{Produce}\ c\ n\ d]\!]^{\emptyset}_{\{c\}} = \mu X.d^n.\underline{\overline{c}}.X$$

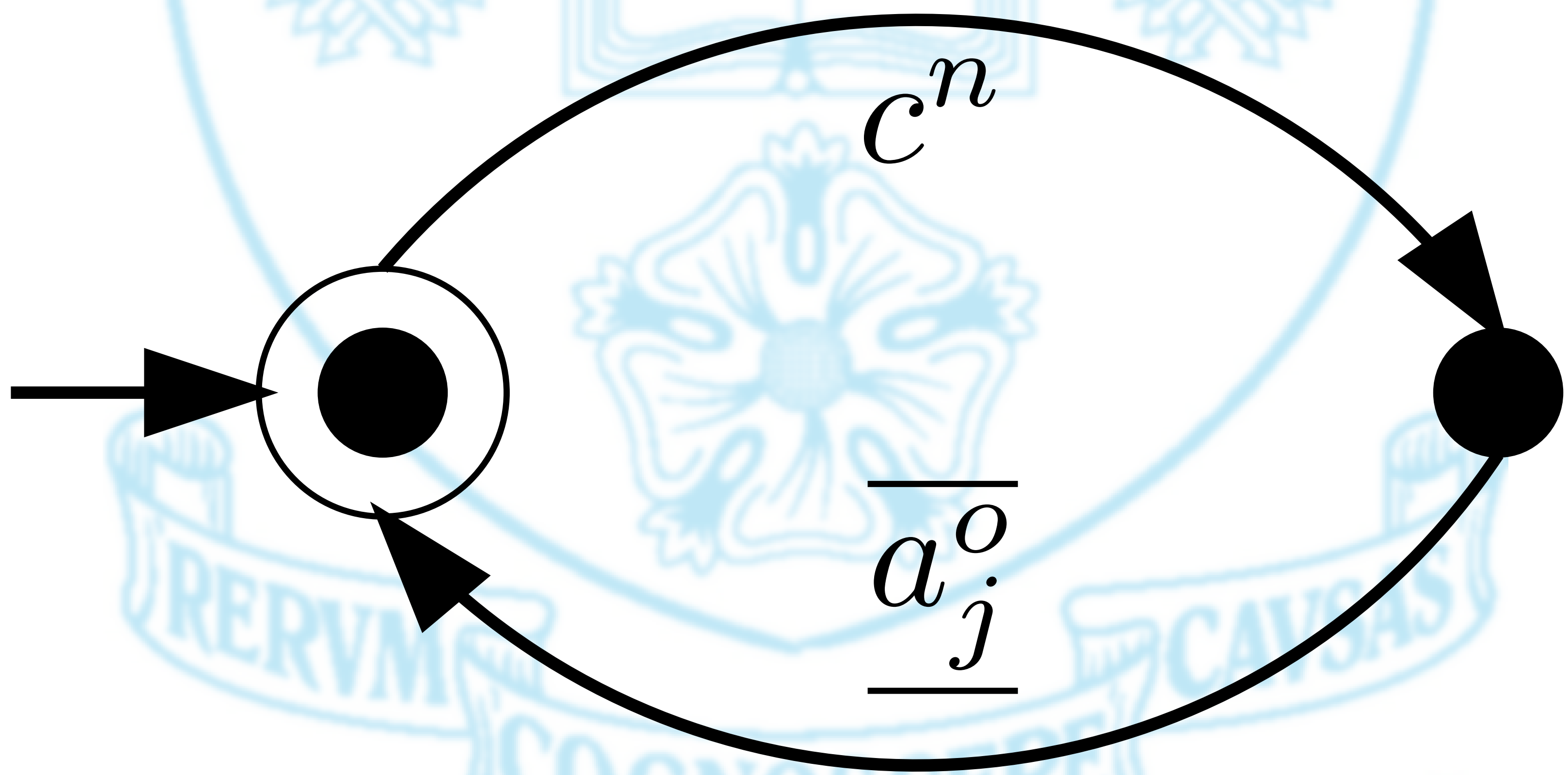


- Within CASheW-s, *Produce* is not a type of performance, rather a type of connection

Connection Semantics

- *Connect* shunts the output of one performance in a composite process, to the input of another.

$$[\![\mathbf{Connect}\ n\ c\ o\ a\ j]\!] = \mu X.c^n.\underline{\overline{a_j^o}}.X$$



Composite Process Semantics

- Defined in terms of a top-level *Governor* process, and in the case of unbounded child-performances an inductively defined context-based composition semantics, which pair a *Scheduler* with the performance semantics.

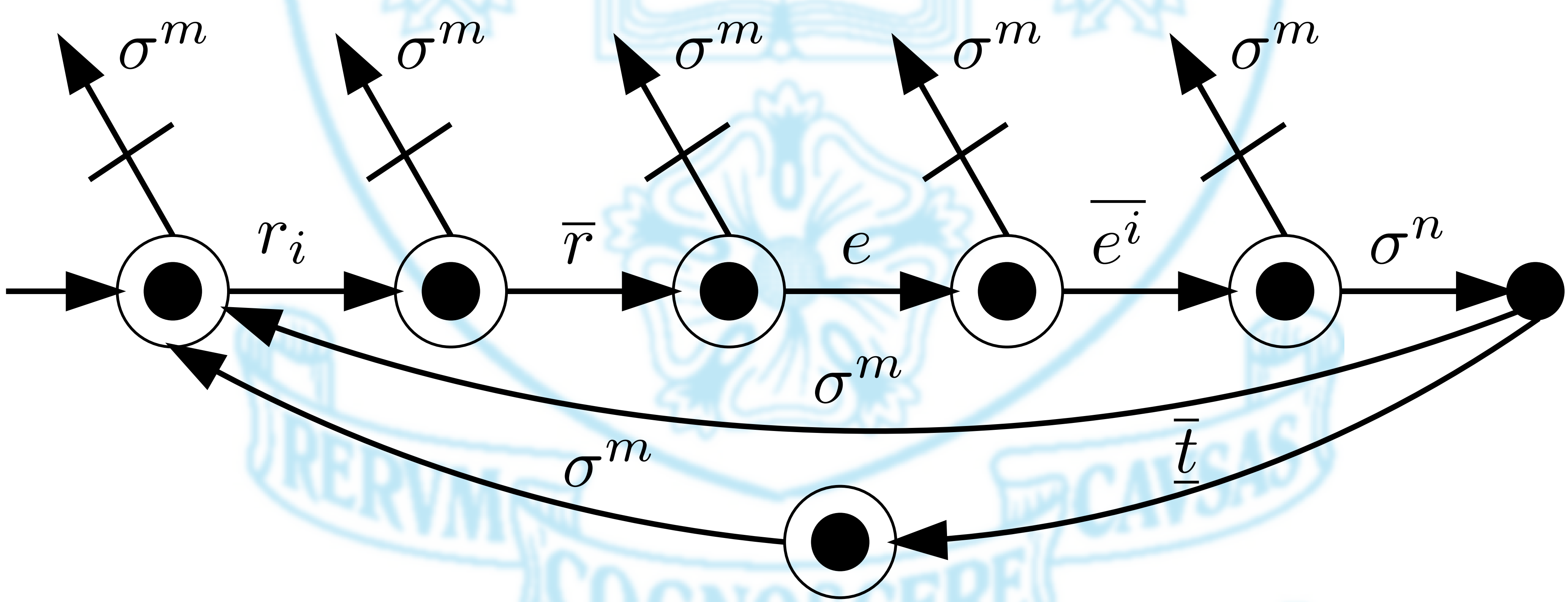
$$^{m}[\![\textbf{Sequence}\ Q]\!]^{A}_{C} = ^{m} [\![^{seq}Q]\!]^{A}_{C} / \sigma^{m} \setminus t$$

$$^{m}[\![\textbf{SplitJoin}\ Q]\!]^{A}_{C} = (^{m}[\![^{sj}Q]\!]^{A}_{C} \mid \mu X.\sigma^{m}.\overline{r}.e.\sigma^{m}.\sigma^{m}.X) /\!\!/ \sigma^{m}$$

$$^{m}[\![\textbf{AnyOrder}\ Q]\!]^{A}_{C} = ^{m} [\![^{any}Q]\!]^{A}_{C} / \sigma^{m} \setminus t$$

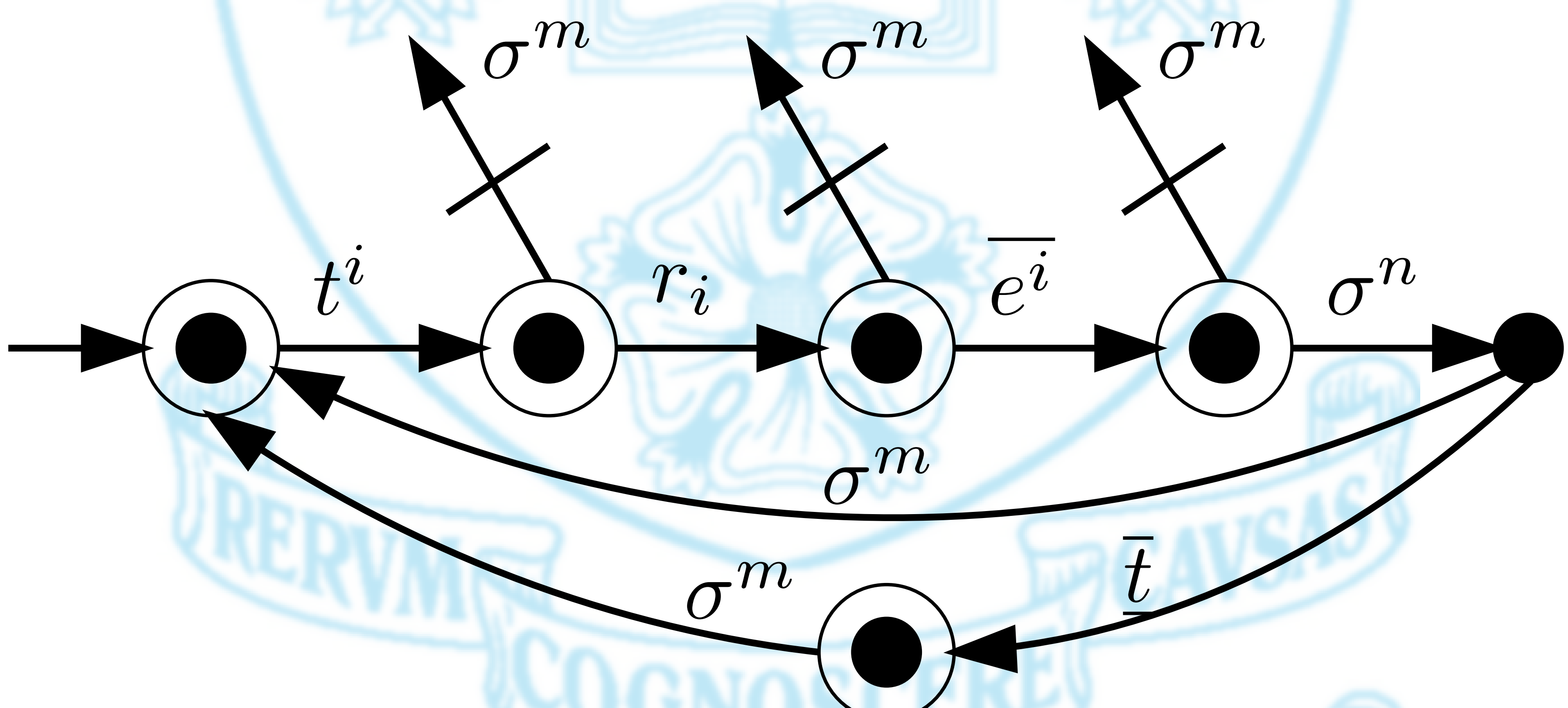
Sequence Semantics
Base Case

$$
\begin{aligned}
&{}^{m}[\![{}^{seq}\mathbf{Perform}\ n\ p\ U\ V]\!]^{A}_{C} = \\
&\quad ({}^{n}[\![\mathbf{Perform}\ n\ p\ U\ V]\!]^{A^n}_{C^n}[e \mapsto e^i, r \mapsto r^i] \mid \\
&\quad \mu X.\underline{r^i.\overline{r}.e.\overline{e^i}.\sigma^n}_{\sigma^m}\lfloor \underline{\overline{t}}.\sigma^m.X\rfloor \sigma^m(X))/\sigma^n \setminus \{r^i, e^i\}
\end{aligned}
$$



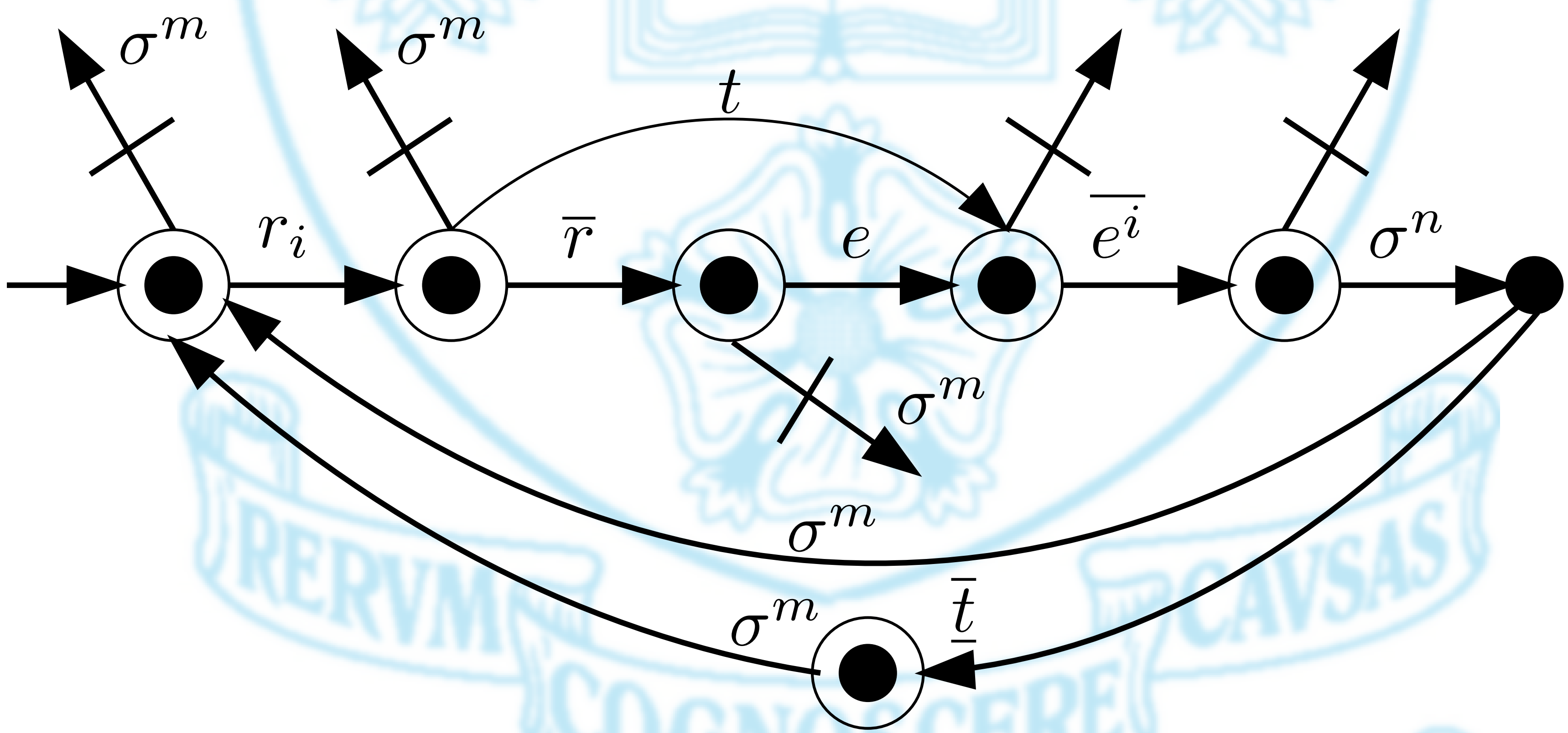
Sequential Composition Semantics General Case

$$
\begin{aligned}
&{}^{m}[\![{}^{seq}(Q); \textbf{Perform}\ n\ p\ U\ V]\!]^{A^Q \cup A^n}_{C^Q \cup C^n} = \\
&\quad ({}^{n}[\![\textbf{Perform}\ n\ p\ U\ V]\!]^{A^n}_{C^n}[e \mapsto e^i, r \mapsto r^i] \mid {}^{m}[\![{}^{seq}Q]\!]^{A^Q}_{C^Q}[t \mapsto t^i] \mid \\
&\quad \mu X.t^i.\underline{r^i.\overline{e^i}.\sigma^n}_{\sigma^m}\lfloor \overline{t}.\sigma^m.X \rfloor \sigma^m(X))/\sigma^n \setminus \{r^i, e^i\}
\end{aligned}
$$



AnyOrder Composition Semantics Base Case

$$
\begin{aligned}
&{}^{m}[\![{}^{any}\mathbf{Perform}\ n\ p\ U\ V]\!]^{A}_{C} = \\
&\quad ({}^{m}[\![{}^{any}\mathbf{Perform}\ n\ p\ U\ V]\!]^{A}_{C}[e \mapsto e^{i}, r \mapsto r^{i}] \mid \\
&\quad \mu \underline{X}.\underline{r^{i}}_{\sigma^{m}}.(\underline{\overline{r}.e.\overline{e^{i}}.\sigma^{n}}_{\sigma^{m}}.\lfloor \underline{\overline{t}}.\sigma^{m}.X \rfloor \sigma^{m}(X) + \\
&\quad \underline{t.\overline{e^{i}}.\sigma^{n}}_{\sigma^{m}}.\lfloor \underline{\overline{t}}.\sigma^{m}.X \rfloor \sigma^{m}(X))) \setminus \{e^{i}, r^{i}\}/\sigma^{n}
\end{aligned}
$$



AnyOrder Composition Semantics General Case

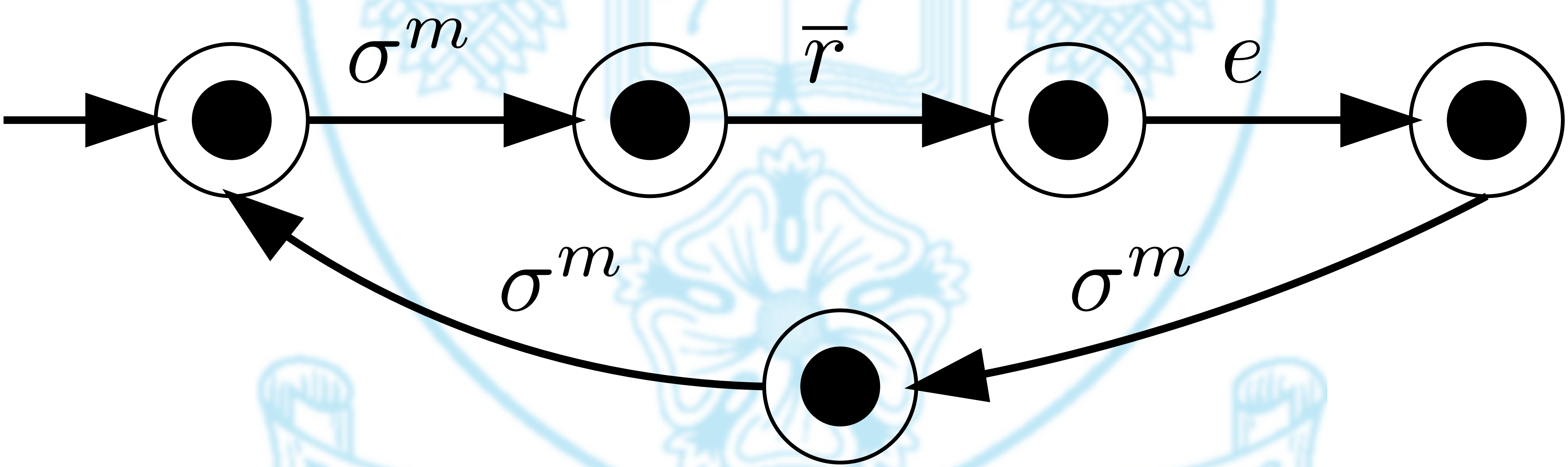
$$
\begin{aligned}
&{}^{m}[\![^{any}(Q); \mathbf{Perform}\ n\ p\ U\ V]\!]^{A^Q \cup A^n}_{C^Q \cup C^n} = \\
&\qquad {}^{m}[\![^{any}\mathbf{Perform}\ n\ p\ U\ V]\!]^{A^n}_{C^n} \mid {}^{m}[\![^{any}Q]\!]^{A^Q}_{C^Q}
\end{aligned}
$$

- We use this induction in all cases to define the semantics for the general case where all performances are handled in the same way.

Split/SplitJoin Process Semantics (Governor)

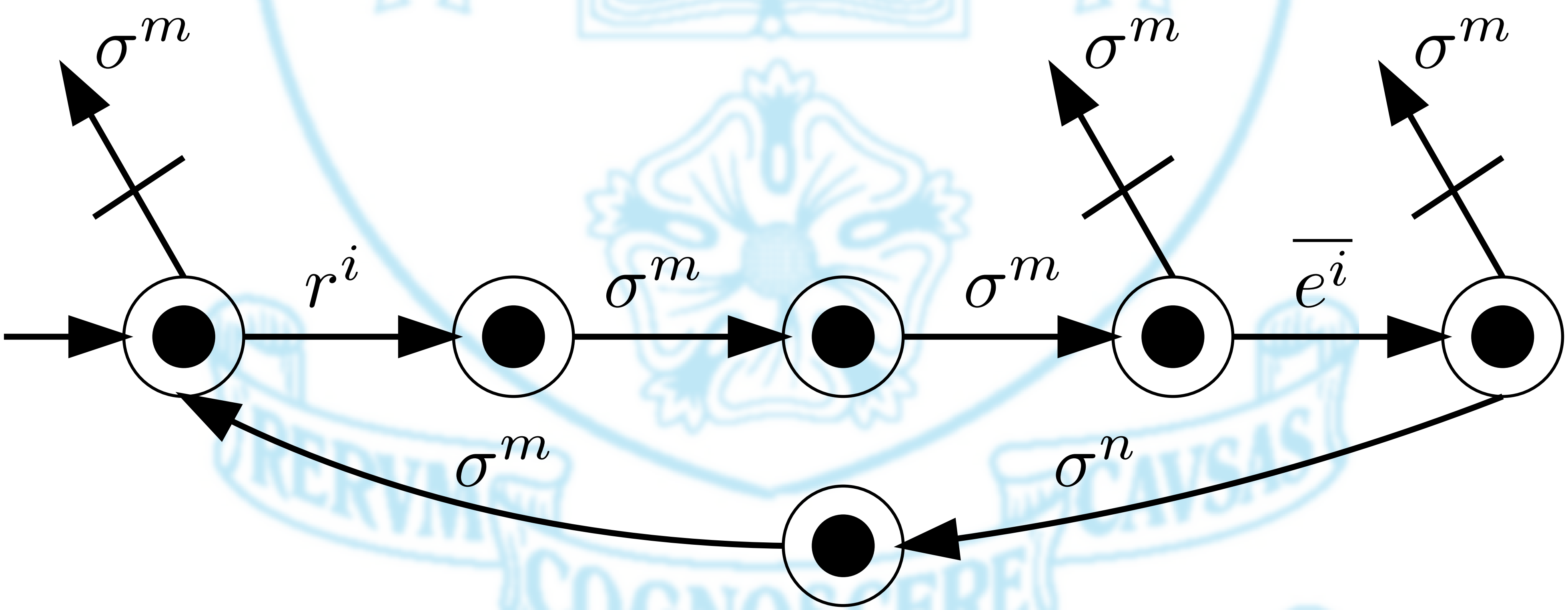
$$^m[\![\textbf{SplitJoin}\ Q]\!]^A_C = (^m[\![^{sj}Q]\!]^A_C \mid \mu X.\sigma^m.\overline{r}.e.\sigma^m.\sigma^m.X) /\!/ \sigma^m$$

$$^m[\![\textbf{Split}\ Q]\!]^A_C = (^m[\![^{split}Q]\!]^A_C \mid \mu X.\sigma^m.\overline{r}.e.\sigma^m.\sigma^m.X) /\!/ \sigma^m$$



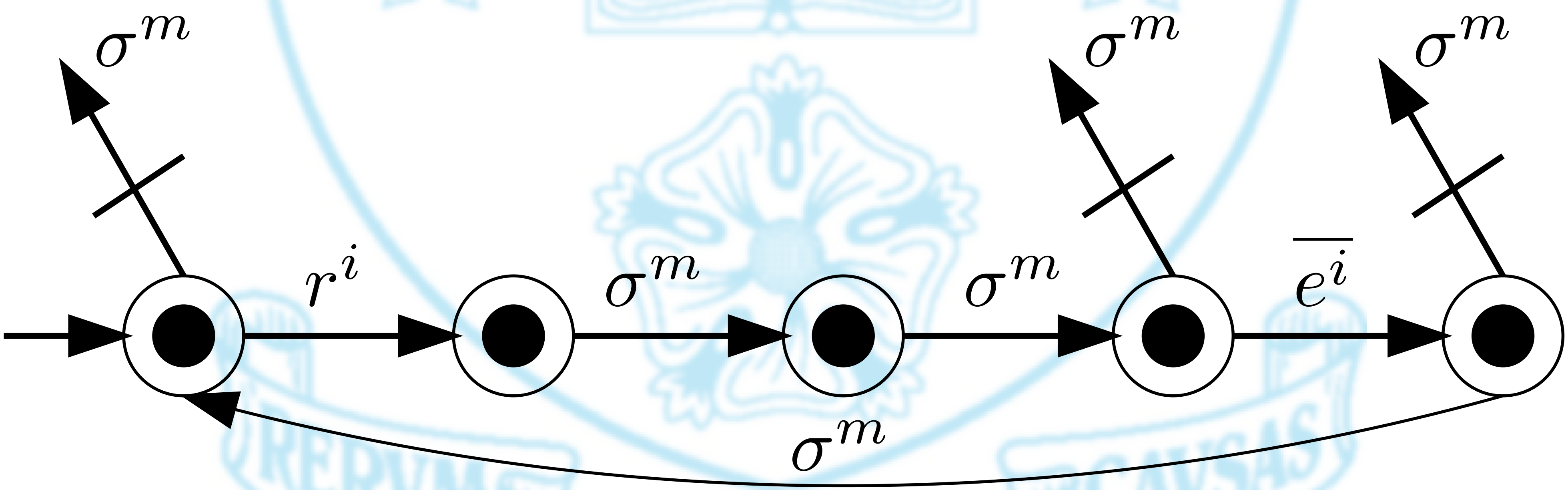
SplitJoin Composition Semantics

$$
\begin{aligned}
&{}^{m}[\![{}^{sj}\mathbf{Perform}\ n\ p\ U\ V]\!]^{A}_{C} = \\
&\quad ({}^{m}[\![{}^{sj}\mathbf{Perform}\ n\ p\ U\ V]\!]^{A}_{C}[e \mapsto e^{i}, r \mapsto r^{i}] \mid \\
&\quad \mu X.\underline{r^{i}}_{\sigma^{m}}.\sigma^{m}.\sigma^{m}.\underline{\overline{e^{i}}.\sigma^{n}}_{\sigma^{m}}.\sigma^{m}.X) \setminus \{e^{i}, r^{i}\}/\sigma^{n}
\end{aligned}
$$



Split Composition Semantics

$$
\begin{aligned}
&{}^{m}[\![{}^{split}\mathbf{Perform}\ n\ p\ U\ V]\!]^{A}_{C} = \\
&\quad ({}^{m}[\![{}^{split}\mathbf{Perform}\ n\ p\ U\ V]\!]^{A}_{C}[e \mapsto e^{i}, r \mapsto r^{i}] \mid \\
&\quad \mu X.\underline{r^{i}}_{\sigma^{m}}.\sigma^{m}.\sigma^{m}.\underline{\overline{e^{i}}}_{\sigma^{m}}.\sigma^{m}.X) \setminus \{e^{i}, r^{i}\}/\sigma^{n}
\end{aligned}
$$



- Split is our primary motivation for clock ticks not bound by maximal progress

Next Step : Haskell Implementation

- We already have an implementation of the CaSHew-NUtS Process Calculus in Haskell, the next step is to define semantics for mapping OWL-S to this representation.
- The Haskell implementation allows the calculus to be grounded in IO operations, enabling Web-Service invocation.
- This can then be combined with our HAIFA interoperability kit to enable orchestration.

Conclusion

- We have presented a timed process calculus semantics for OWL-S, which we will shortly be using to build an orchestration engine.
- We predict that this approach to providing operational semantics can be applied to other work-flow languages, allowing a single engine to be able handle heterogeneous orchestration.
- All of this will be combined with the safety of Haskell, to build reliable, predictable workflows.

More to come soon...

Basic CCS Rules

Act $\dfrac{}{\alpha.E \xrightarrow{\alpha} E}$ Sum1 $\dfrac{E \xrightarrow{\alpha} E'}{E + F \xrightarrow{\alpha} E'}$ Sum2 $\dfrac{F \xrightarrow{\alpha} F'}{E + F \xrightarrow{\alpha} F'}$

Com1 $\dfrac{E \xrightarrow{\alpha} E'}{E \mid F \xrightarrow{\alpha} E' \mid F}$ Com2 $\dfrac{F \xrightarrow{\alpha} F'}{E \mid F \xrightarrow{\alpha} E \mid F'}$

Com3 $\dfrac{E \xrightarrow{a} E', F \xrightarrow{\overline{a}} F'}{E \mid F \xrightarrow{\tau} E' \mid F'}$ Res $\dfrac{E \xrightarrow{\gamma} E'}{E \setminus a \xrightarrow{\gamma} E' \setminus a} \gamma \notin \{a, \overline{a}\}$

Rec $\dfrac{E \xrightarrow{\gamma} E'}{\mu X.E \xrightarrow{\gamma} E'\{\mu X.E/X\}}$

CaSE Additions to CCS

Idle $\dfrac{}{\mathbf{0} \xrightarrow{\sigma_1} \mathbf{0}}$ Patient $\dfrac{}{a.E \xrightarrow{\sigma_1} a.E}$ Stall $\dfrac{}{\Delta_\sigma \xrightarrow{\rho_1} \Delta_\sigma}$ 1

TO1 $\dfrac{}{\lfloor E \rfloor \sigma(F) \xrightarrow{\sigma_i} F}$ a TO2 $\dfrac{E \xrightarrow{\gamma} E'}{\lfloor E \rfloor \sigma(F) \xrightarrow{\gamma} E'}$ 2

TO3 $\dfrac{E \xrightarrow{\sigma_i} E'}{\lfloor E \rfloor \sigma(F) \xrightarrow{\sigma_0} E'}$

where: 1) $\rho \neq \sigma$
2) $\nexists i \cdot \gamma = \sigma_i$

and: a) $i = 0$ if $\tau \in \mathcal{IA}(E)$, 1 otherwise
b) $k = 0$ if $\tau \in \mathcal{IA}(E \mid F)$, 1 otherwise
c) $\sigma_1 \notin \mathcal{IA}(E)$
d) $\nexists i \cdot \sigma_i \in \mathcal{IA}(E)$

CaSE Additions (cont)

$$\mathsf{STO1}\ \frac{}{\lfloor E \rfloor \sigma(F) \xrightarrow{\sigma_i} F}\ a \qquad \mathsf{STO2a}\ \frac{E \xrightarrow{\alpha} E'}{\lfloor E \rfloor \sigma(F) \xrightarrow{\alpha} E'}\ 2$$

$$\mathsf{STO3}\ \frac{E \xrightarrow{\sigma_i} E'}{\lfloor E \rfloor \sigma(F) \xrightarrow{\sigma_0} E'} \qquad \mathsf{STO2b}\ \frac{E \xrightarrow{\rho_i} E'}{\lfloor E \rfloor \sigma(F) \xrightarrow{\rho_i} E'}\ 1$$

$$\mathsf{Com4}\ \frac{E \xrightarrow{\sigma_i} E' \quad F \xrightarrow{\sigma_j} F'}{E \mid F \xrightarrow{\sigma_{i \cdot j \cdot k}} E' \mid F'}\ b$$

where: 1) $\rho \neq \sigma$
2) $\nexists i \cdot \gamma = \sigma_i$

and: a) $i = 0$ if $\tau \in \mathcal{IA}(E)$, 1 otherwise
b) $k = 0$ if $\tau \in \mathcal{IA}(E \mid F)$, 1 otherwise
c) $\sigma_1 \notin \mathcal{IA}(E)$
d) $\nexists i \cdot \sigma_i \in \mathcal{IA}(E)$

CaSE Additions (cont)

$$\text{Hid1} \quad \frac{P \xrightarrow{\sigma_1} P'}{P/\sigma \xrightarrow{\tau} P'/\sigma} \qquad \text{Hid2} \quad \frac{P \xrightarrow{\alpha} P'}{P/\sigma \xrightarrow{\alpha} P'/\sigma}$$

$$\text{Hid3} \quad \frac{P \xrightarrow{\rho_i} P'}{P/\sigma \xrightarrow{\rho_i} P'/\sigma} \, 1, c$$

where:	1) $\rho \neq \sigma$	and:	a) $i = 0$ if $\tau \in \mathcal{IA}(E)$, 1 otherwise
	2) $\nexists i \cdot \gamma = \sigma_i$		b) $k = 0$ if $\tau \in \mathcal{IA}(E \mid F)$, 1 otherwise
			c) $\sigma_1 \notin \mathcal{IA}(E)$
			d) $\nexists i \cdot \sigma_i \in \mathcal{IA}(E)$

CaSHew-NUtS

$$\mathsf{UHid1}\ \frac{P \xrightarrow{\sigma_i} P'}{P /\!\!/ \sigma \xrightarrow{\tau} P' /\!\!/ \sigma} \qquad \mathsf{UHid2}\ \frac{P \xrightarrow{\alpha} P'}{P /\!\!/ \sigma \xrightarrow{\alpha} P' /\!\!/ \sigma}$$

$$\mathsf{UHid3}\ \frac{P \xrightarrow{\rho_i} P'}{P /\!\!/ \sigma \xrightarrow{\rho_i} P' /\!\!/ \sigma}\ 1, d$$

where: 1) $\rho \neq \sigma$
2) $\nexists i \cdot \gamma = \sigma_i$

and: a) $i = 0$ if $\tau \in \mathcal{IA}(E)$, 1 otherwise
b) $k = 0$ if $\tau \in \mathcal{IA}(E \mid F)$, 1 otherwise
c) $\sigma_1 \notin \mathcal{IA}(E)$
d) $\nexists i \cdot \sigma_i \in \mathcal{IA}(E)$

The University Of Sheffield.