1 LTS; ACP

LTS and Process Graphs Both specifications and implementations could be represented by <u>*models of concurrency*</u>, for example *labelled transition systems (LTS)* or *process graphs.*

Definition 1.1 (Process Graph) *A process graph is a triple $(S, I, \rightarrow)$ such that:*

- *S a set of states;*
- *$I \in S$ an initial state;*
- *$\rightarrow$ a set of triples (s, a, t) each describing a (named) relation $S \to S$:*
  - *$s, t \in S$;*
  - *$a \in Act$ – a set of actions.*

Definition 1.2 (LTS) *Same as process graph, except without an initial state. Sometimes used synonymously with process graphs bc. mathematicians are evil.*

Alternatively, one may use *process algebraic expressions* to formally represent spec.s and impl.s, for example using *CCS (Calculus of Communicating Systems)*, *CSP (Communicating Sequential Processes)*, and *ACP (Algebra of Communicating Processes)*. <u>Each semantics is of different expressive power.</u>

ACP Define the set of operations:

- ε (successful termination – ACP_ε extension).
- δ (deadlock).
- a (action constant) for each action $a \in Act$.

  Each a describe a **visible action** – $\tau \notin Act$;
- $P \cdot Q$ (sequential composition between processes P, Q)
- $P + Q$ (summation / choice / alternative composition);
- $P || Q$ (parallel composition).
- $\partial_H(P)$ (restriction / encapsulation).

  Given set of (visible) actions H, this removes $\forall a \in H$ in P.

  Practically this is often used after defining $\gamma(a, b)$ to enforce sync – via removing non-synced $a.b$ or $b.a$ behaviors;
- $\tau_I(P)$ (abstraction – ACP_τ extension).

  Given set of (visible) actions I, this converts $\forall a \in I$ into τ in P.

  A τ action is **non-observable** – this will be significant for describing traces & equivalence relations.
- $\gamma : A \times A \to A$ (partial communication function).

  For example, $\gamma(a, b)$ defines new (synchronized) visible action alongside a, b.

We further define the following transition rules (omitting commutative equivalents). First, transition rules for basic process algebra wrt. termination, sequential composition, and choice:

$$\frac{}{a \xrightarrow{a} \varepsilon} \qquad \frac{a \xrightarrow{a} \varepsilon}{a + b \xrightarrow{a} \varepsilon} \qquad \frac{a \xrightarrow{a} \varepsilon}{a \cdot b \xrightarrow{a} b}$$

$$\frac{a \xrightarrow{a} a'}{a + b \xrightarrow{a} a'} \qquad \frac{a \xrightarrow{a} a'}{a \cdot b \xrightarrow{a} a' \cdot b}$$

Then, for parallel processes which may or may not communicate:

$$\frac{a \xrightarrow{a} \varepsilon}{a||b \xrightarrow{a} b} \qquad \frac{a \xrightarrow{a} a'}{a||b \xrightarrow{a} a'||b}$$

$$\frac{a \xrightarrow{a} \varepsilon \quad b \xrightarrow{b} \varepsilon}{a||b \xrightarrow{\gamma(a,b)} \varepsilon} \qquad \frac{a \xrightarrow{a} a' \quad b \xrightarrow{b} \varepsilon}{a||b \xrightarrow{\gamma(a,b)} a'}$$

$$\frac{a \xrightarrow{a} \varepsilon \quad b \xrightarrow{b} b'}{a||b \xrightarrow{\gamma(a,b)} b'} \qquad \frac{a \xrightarrow{a} a' \quad b \xrightarrow{b} b'}{a||b \xrightarrow{\gamma(a,b)} a'||b'}$$

Furthermore, for encapsulation ∂_H:

$$\frac{a \xrightarrow{x} \varepsilon}{\partial_H(a) \xrightarrow{x} \varepsilon}\, x \notin H \qquad \frac{a \xrightarrow{x} a'}{\partial_H(a) \xrightarrow{x} \partial_H(a')}\, x \notin H$$

This is to say, $\partial_H(a)$ can execute all transitions of a that are also not in H.

Finally, deadlocks **does not display any behavior** – that is, a δ process cannot transition to any other states no matter what (though obviously as a constituent part of e.g., a parallel process the other concurrent constituent can still run).

Background 1.1 (commutativity)

$$f(a, b) = f(b, a) \iff f \text{ commutative}$$

Background 1.2 (associativity)

$$(a \circ b) \circ c = a \circ (b \circ c) \iff \circ \text{ associative}$$

Background 1.3 (distributivity)

$$f(x, a \circ b) = f(x, a) \circ f(x, b) \iff f \text{ distributes over } \circ$$

Background 1.4 (isomorphism) *An isomorphism describes a* ***bijective homomorphism****:*

- ***Homomorphism*** *describes a* ***structure-preserving*** *map between two algebraic* ***structures*** *of the same* ***type****:*
  - ***Algebraic structure*** *describes a set with additional properties – e.g., an additive group over $\mathbb{N}$, a ring of integers modulo x, etc.*
  - *Two structures of the same* ***type*** *refers to structures with the same name – e.g., two groups, two rings, etc.*
  - *A* ***structure-preserving*** *map f between two structures intuitively describes a structure such that, for properties $p \in X$, $q \in Y$ between same-type structures X, Y, any tuples $X^n \in p$ accepted by p (e.g., $3 + 5 = 8 \implies (3, 5, 8) \in \mathbb{R}.(+)$) satisfies $\mathsf{map}(f, X^n) \in q$.*
- ***Bijection*** *describes a 1-to-1 correspondence between elements of two sets – i.e., invertible.*

2 Semantic Equivalences

Background 2.1 (lattice) *A **lattice** describes a real coordinate space $\mathbb{R}^n$ that satisfies:*

- *Addition / subtraction between two points always produce another point in lattice – i.e., closed under addition / subtraction.*
- *Lattice points are separated by bounded distances in some range* $(0, \max]$.

Define a lattice over which *semantic equivalence relations* for spec. and impl. verification is defined.

Definition 2.1 (discrimination measure) *One equivalence relation $\equiv$ is **finer** / **more discriminating** than another $\sim$ if each $\equiv$-eq. class is a subset of a $\sim$-eq. class. In other words,*

$$\begin{aligned} p \equiv q &\implies p \sim q \\ \iff &\equiv \text{ finer than } \sim \end{aligned}$$