CSP Chapter 3: Nondeterminism

Andrzej Filinski
DIKU

Extreme Multiprogramming 6
December 6, 2013

Overview

- Today: nondeterministic processes.
  - Conceptually simple addition, massive ramifications.
  - Fortunately, all formal laws introduced so far remain valid...
  - ... but *informal* reasoning gets far more dangerous!
- Nondeterminism: exactly same actions and offers from environment may result in different behavior by process.
  - Usefulness of testing greatly diminished.
  - Even more need for "correctness by design".
- Two sources of nondeterminism in CSP
  - Explicit: manifestly unpredictable choices.
  - Implicit: asynchronous *concealed* events, in connection with normally deterministic choice.
- Both ultimately modeled by a single construct.

Essence of nondeterminism: *internal choice*

- **Notation:** $P \sqcap Q$ ("P or Q").
  - $\alpha(P \sqcap Q) = \alpha P = \alpha Q$.
- Process that can behave either like P or like Q; but the process itself (not the environment) decides which one.
- **Examples:**
  - $VME \triangleq coin \to (coke \to VME \mid sprite \to VME)$.
    External choice: *customer* decides on product.
  - $VMI \triangleq coin \to ((coke \to VMI) \sqcap (sprite \to VMI))$.
    Internal choice: *machine* decides on product.
- Note: $traces(VME) = traces(VMI)$!
- With nondeterminism in the picture, $traces(P)$ no longer fully describes behavior of P; will also need to consider what P may *refuse* to do.

Reasoning about nondeterminism

- Idempotence: $P \sqcap P = P$
- Commutativity: $P \sqcap Q = Q \sqcap P$
- Associativity: $(P \sqcap Q) \sqcap R = P \sqcap (Q \sqcap R)$.
- *No* neutral element: no *fixed* process $MAGIC$ (with $\alpha MAGIC = \alpha P$), such that $P \sqcap MAGIC = P$.
- Prefix distributivity: $x \rightarrow (P \sqcap Q) = (x \rightarrow P) \sqcap (x \rightarrow Q)$.
  - More generally: $(v\colon B \rightarrow (P(v) \sqcap Q(v)) = (v\colon B \rightarrow P(v)) \sqcap (v : B \rightarrow Q(v))$.
  - Can't observe exactly *when* an internal choice is made.
- Concurrency distributivity: $(P \sqcap Q) \parallel R = (P \parallel R) \sqcap (Q \parallel R)$.
- But difference between choosing *repeatedly* or *once*:
  - $VMI = \mu X.((coin \rightarrow coke \rightarrow X) \sqcap (coin \rightarrow sprite \rightarrow X))$
  - $VMI' = (\mu X.coin \rightarrow coke \rightarrow X) \sqcap (\mu X.coin \rightarrow sprite \rightarrow X))$
  - $traces(VMI') \subsetneq traces(VMI)$.

More about internal choice

- Internal choice need not be *fair*:
  - $VMI \triangleq coin \to ((coke \to VMI) \sqcap (sprite \to VMI))$ can decide to serve only *coke*s forever.
  - For communication: can ignore a channel forever.
- Internal choice need not even be *responsive*:
  - Let $CUSTC \triangleq coin \to coke \to CUSTC$.
    Then $VMI \parallel CUSTC$ can deadlock.
  - So can $VMI \parallel CUSTI$, where
    $CUSTI \triangleq coin \to ((coke \to CUSTI) \sqcap (sprite \to CUSTI))$.
  - But $VMI \parallel CUSTE = VMI$, if
    $CUSTE \triangleq coin \to (coke \to CUSTE \mid sprite \to CUSTE)$,

General choice

- Hybrid between internal and external choice.
- **Notation:** $P \Box Q$. ("P alt Q"?).
  - $\alpha(P \Box Q) = \alpha P = \alpha Q$.
- Like $P \sqcap Q$, but will never *insist* on a first event that environment is unwilling to participate in.
  - $(x \to P) \Box (y \to Q) = (x \to P \mid y \to Q)$, if $x \neq y$.
  - $(x \to P) \Box (x \to Q) = (x \to P) \sqcap (x \to Q) = x \to (P \sqcap Q)$.
- **Examples:**
  - $VME \triangleq coin \to ((coke \to VME) \Box (sprite \to VME))$
  - $VMI \triangleq (coin \to coke \to VMI) \Box (coin \to sprite \to VMI)$

More about general choice

- **Caution:** Don't anthropomorphize CSP processes. (They don't like it...) Still:
- $P \,\Box\, Q$ makes good-faith effort to avoid *immediate* deadlock.
  - $VME \triangleq coin \to ((coke \to VME) \,\Box\, (sprite \to VME))$
  - $VME \parallel (coin \to sprite \to P) = coin \to sprite \to (VME \parallel P)$.
- But not *prescient*: cannot pick right branch to avoid *future* deadlock:
  - $VMI \triangleq (coin \to coke \to VMI) \,\Box\, (coin \to sprite \to VMI))$
  - $VMI \parallel (coin \to sprite \to P) \neq coin \to sprite \to (VMI \parallel P)$.
- Sometimes useful to also consider *angelic* and *demonic* nondeterminism:
  - $P \sqcap^{ang} Q$ chooses so as to avoid future deadlock (or other undesirable behavior), if at all possible.
  - $P \sqcap^{dem} Q$ chooses so as to cause future deadlock (or other undesirable behavior), if at all possible.
  - Implementable in principle (though very inefficiently), but only if environment is fully specified as another CSP process.

Reasoning about general choice

- Idempotence, associativity, commutativity, like for $\sqcap$:
  $P \Box P = P$, $P \Box Q = Q \Box P$, $(P \Box Q) \Box R = P \Box (Q \Box R)$.
- Now also have a neutral element: $P \Box STOP_{\alpha P} = P$.
- Law for general choice between parameterized choices:
  $(v\colon A \to P(v)) \Box (v\colon B \to Q(v)) = (v\colon A \cup B \to R(v))$,
  where

$$R(x) = \begin{cases} P(x) & \text{if } x \in A, x \notin B \\ Q(x) & \text{if } x \in B, x \notin A \\ P(x) \sqcap Q(x) & \text{if } x \in A, x \in B \end{cases}$$

- But note: with nondeterminism, not all processes can be expressed as particular instances of parameterized choice.
  - Also need laws like $(P \sqcap Q) \Box R = (P \Box R) \sqcap (Q \Box R)$.
- **Caution:** $(P \sqcap Q) \parallel R = (P \parallel R) \sqcap (Q \parallel R)$, but no such law as $(P \Box Q) \parallel R = (P \parallel R) \Box (Q \parallel R)$.
  - Must resolve $\Box$ into either $\sqcap$ or $|$ before reasoning can proceed.

POP QUIZ: General choice

- Consider the following process definitions:

$$
\begin{aligned}
VM &\triangleq (coin \to coke \to VM) \,\Box\, (coin \to sprite \to VM) \,\Box\\
&\qquad (water \to VM)
\end{aligned}
$$

$$
\begin{aligned}
C1 &\triangleq coin \to sprite \to C1\\
C2 &\triangleq coin \to ((coke \to C2) \,\Box\, (sprite \to C2))\\
C3 &\triangleq (coin \to ((coke \to C3) \,\Box\, (sprite \to C3))) \sqcap (water \to C3)\\
C4 &\triangleq coin \to ((coke \to C4) \,\Box\, (sprite \to C4) \,\Box\, (water \to C4))
\end{aligned}
$$

All alphabets are $\{coin, coke, sprite, water\}$

- What is the behavior of each of the following? Can they deadlock? Can they be written more simply?
  - $VM \parallel C1$
  - $VM \parallel C2$
  - $VM \parallel C3$
  - $VM \parallel C4$

Refusals

- Let $P = coke \rightarrow STOP_A$, $Q = sprite \rightarrow STOP_A$.
  - $traces(P \mathbin{\Box} Q) = traces(P \sqcap Q) = \{\langle\rangle, \langle coke\rangle, \langle sprite\rangle\}$
  - But $(P \mathbin{\Box} Q) \parallel P = P$, while $(P \sqcap Q) \parallel P = P \sqcap STOP \neq P$.
- The traces of a nondeterministic process do not fully describe its behavior!
- Trace sets describe everything that a process *might* do (for some sequence of internal choices); also need to know what it might choose *not* to do.
- **Def**: a *refusal* of a process P is a set $C \subseteq \alpha P$ such that P *may* deadlock on its first step, when offered only C by the environment.
  - I.e., if $P \parallel (v : C \rightarrow RUN_{\alpha P}) = STOP \sqcap P'$ for some P'.
- The set of all refusals of P is written $refusals(P)$. Note that this is a *set of sets*.

Refusals (2)

- **Recall:**
  - $\alpha VME = \alpha VMI = \{coin, coke, sprite\}$.
  - $VME \triangleq coin \to ((coke \to VME) \,\square\, (sprite \to VME))$.
  - $VMI \triangleq coin \to ((coke \to VMI) \sqcap (sprite \to VMI))$.
- Then $refusals(VME) = refusals(VMI) = \{\{\}, \{coke\}, \{sprite\}, \{coke, sprite\}\}$.
  - If environment's offer does not include $coin$, system is deadlocked.
- But $refusals(VME \,/\, \langle coin \rangle) = \{\{\}, \{coin\}\}$
  - After a $coin$, the machine will not refuse any offer that includes a product event.
- Whereas $refusals(VMI \,/\, \langle coin \rangle) = \{\{\}, \{coin\}, \{coke\}, \{sprite\}, \{coin, coke\}, \{coin, sprite\}\}$
  - After a $coin$, machine can choose to refuse any offer that does not include both $coke$ and $sprite$.

Reasoning about refusals

- If a process can choose to deadlock when offered a choice, it can certainly also deadlock when offered a smaller choice:
  - If $C' \subseteq C$ and $C \in \mathit{refusals}(P)$, then $C' \in \mathit{refusals}(P)$.
  - In particular, $\{\} \in \mathit{refusals}(P)$ for any P.
- If P is deterministic, it is completely controlled by the environment: cannot refuse anything that it might also have accepted to do:
  - $\mathit{refusals}(P) = \{C \mid \neg\exists x.\, x \in C \wedge \langle x \rangle \in \mathit{traces}(P)\}$, for P deterministic.
- In general, the behavior of a CSP process P is fully determined by
  a. knowing $\mathit{traces}(P)$, and
  b. For every $s \in \mathit{traces}(P)$, knowing $\mathit{refusals}(P \mathbin{/} s)$.
- For a *persistently deterministic* process, (a) is sufficient, because (b) can be determined from it.

Refusal laws

- $\mathit{refusals}(v : B \to P(v)) = \{C \subseteq \alpha P \mid C \cap B = \{\}\}$.
  Parameterized choice can (and will) refuse any offer that has nothing in common with initial menu. Special cases:
  - $\mathit{refusals}(STOP_A) = \{C \mid C \subseteq A\}$
  - $\mathit{refusals}(x \to P) = \{C \mid x \notin C\}$
- $\mathit{refusals}(P \sqcap Q) = \mathit{refusals}(P) \cup \mathit{refusals}(Q)$.
  Internal choice can refuse any offer that *either* component can refuse.
- $\mathit{refusals}(P \,\square\, Q) = \mathit{refusals}(P) \cap \mathit{refusals}(Q)$.
  General choice can refuse any offer that *both* components can refuse.
- $\mathit{refusals}(P \parallel Q) = \{A \cup B \mid A \in \mathit{refusals}(P), B \in \mathit{refusals}(Q)\}$
  Concurrent composition can refuse any offer that is the union of some refusals of each component. (See next slide.)

Example: Refusals of a concurrent composition

- Consider:

$$\begin{aligned} \alpha VM = \alpha CUST &= \{coke, sprite, fanta\} \\ VM &\triangleq (coke \to VM) \,\Box\, (sprite \to VM) \\ CUST &\triangleq (coke \to CUST) \sqcap (fanta \to CUST) \end{aligned}$$

- Then, using law for $refusals(P \parallel Q)$,

$$\begin{aligned} refusals(VM) &= \{\{\}, \{fanta\}\} \\ refusals(CUST) &= \{\{\}, \{sprite\}, \{coke\}, \{fanta\}, \\ &\quad\ \{coke, sprite\}, \{sprite, fanta\}\} \\ refusals(VM \parallel CUST) &\ni \{fanta\} \cup \{coke, sprite\} \\ &\quad = \{coke, sprite, fanta\} \end{aligned}$$

- So system *can* deadlock on first step. Also seen by:

$$\begin{aligned} VM \parallel CUST &= VM \parallel ((coke \to CUST) \sqcap (fanta \to CUST)) \\ &= (VM \parallel (coke \to CUST)) \sqcap (VM \parallel (fanta \to CUST)) \\ &= (coke \to (VM \parallel CUST)) \sqcap STOP \end{aligned}$$

Concealment

- Often want to conceal some events from environment.
  - environment need not participate
  - environment can't even observe
- **Notation:** $P \setminus C$ ("P hide C"); do not confuse with $P \,/\, s$.
  - $\alpha(P \setminus C) = (\alpha P) - C$
- Behaves like P, but if P can engage in some event $x \in (\alpha P) \cap C$, x can now happen *silently* and *spontaneously*, without synchronizing with environment.
- Consider processes (all alphabets are $\{coin, coke, sprite\}$):

$$\begin{aligned} VMC &\triangleq coin \to (coke \to VMC \mid sprite \to VMC) \\ CUST &\triangleq coin \to coke \to CUST \end{aligned}$$

- $(VMC \parallel CUST) \setminus \{sprite\} = \mu X : \{coin, coke\}.\, coin \to coke \to X$
- $(VMC \parallel CUST) \setminus \{coin\} = \mu X : \{coke, sprite\}.\, coke \to X$
- $(VMC \parallel CUST) \setminus \{coin, coke\} = \mu X : \{sprite\}.\, X$ (?)

Laws about concealment

- $P \setminus \{\} = P$, $(P \setminus C_1) \setminus C_2 = P \setminus (C_1 \cup C_2)$.
- $STOP_A \setminus C = STOP_{A-C}$.
- $(x \to P) \setminus C = x \to (P \setminus C)$, if $x \notin C$.
  - Generalized: $(v : B \to P(v)) \setminus C = (v : B \to (P(v) \setminus C))$, if $B \cap C = \{\}$.
- $(x \to P) \setminus C = (P \setminus C)$, if $x \in C$.
- $(x \to P \mid y \to Q) \setminus C = (P \setminus C) \sqcap (Q \setminus C)$, if $x, y \in C$.
- $(x \to P \mid y \to Q) \setminus C =^? (P \setminus C) \sqcap (y \to (Q \setminus C))$, if $x \in C, y \notin C$
  - No! Even with *partial* concealment, "$\mid$" (or "$\square$") will not pick a choice that manifestly disagrees with environment's offer.

Partial concealment

- Consider: $VM \triangleq (coin \to coke \to VM \mid water \to VM)$
- $VM \setminus \{coin\}$ can spontaneously commit to serving a $coke$, because $coin$ can happen silently.
- But don't want $VM \setminus \{coin\}$ to deadlock system (commit to $water$) when environment only wants to engage in $coke$.
- Yet with plausible-looking "law",

  $$(x \to P \mid y \to Q) \setminus \{x\} = (P \setminus \{x\}) \sqcap (y \to (Q \setminus \{x\}))$$

  we would get

  $$VM \setminus \{coin\} =^{?} (coke \to (VM \setminus \{coin\})) \sqcap (water \to (VM \setminus \{coin\}))$$

  which can refuse $\{coke\}$.
- Correct law: retain external choice, $(x \to P \mid y \to Q) \setminus \{x\} = (P \setminus \{x\}) \sqcap ((P \setminus \{x\}) \,\Box\, (y \to (Q \setminus \{x\})))$.
- Then we get, in particular, $VM \setminus \{coin\} = \mu X.(coke \to X) \sqcap (coke \to X \mid water \to X)$.

Divergence

- Normally, the purpose of a process is to interact with its environment.
- Can conceal some events that the environment cannot legitimately be interested in.
- But what if we hide so many events that process (or process system) needs no longer interact with environment at all?
- $(VM \parallel CUST) \setminus \{coin, coke, sprite\} = ?$.
- In practice: process that spins uselessly, burning arbitrary amounts of CPU time, without interacting with any other process.
  - Particularly bad for cooperative multithreading.
  - "Livelock".

Divergence, continued

- $(\mu X.coke \rightarrow X) \setminus \{coke\} = \mu X.X = \mathit{MYSTERY}$, where $\mathit{MYSTERY} \triangleq \mathit{MYSTERY}$. (Note: Recursion not guarded!)
- **Recall:** our theory of process definitions had the property that any recursive definition determined a *unique* process.
- But what behavior does $\mathit{MYSTERY}$ specify? *Every* candidate process satisfies that equation.
  - In most sequential/deterministic languages, it would be simply an infinite loop.
  - But in CSP, $\mu X : A.\, X = \mathit{CHAOS}_A$, the *least determined* process: may do anything (within A!), may refuse anything.
  - (In practice, *CHAOS* would most likely just loop silently; but for the purpose of reasoning about it, we assume the worst.)
- Even *potential* divergence is bad: $\mu X.(coke \rightarrow \mathit{STOP}) \sqcap X$ can still choose to just spin, but at least doesn't have to.
- Being equivalent to a (potentially) diverging process is almost always a sign of a bug in the original process definition.

Pictures (transition diagrams) revisited

- Recall pictorial representation of processes as (deterministic) automata:
  - nodes: process states
  - labeled edges: events
  - single unlabeled edges: recursion unfolding.
- Concealment operation just corresponds to erasing labels on edges.
  - Can now have *multiple* unlabeled outgoing edges from a node.
  - Nondeterminism!
- Can use CSP laws to *normalize* graph: every node is either an external choice node (all outgoing edges are labelled and distinct) or internal choice (all outgoing edges are unlabelled).
- **Note:** concealment may turn labeled self-loop into unlabeled self-loop: divergence.

Interleaving

- **Notation:** $P \interleave Q$; "P interleave Q".
  - Unlike $\parallel$, has $\alpha(P \interleave Q) = \alpha P = \alpha Q$.
  - Mnemonic: concurrent composition with an interaction barrier.
- Behaves like $P \parallel Q$, but P cannot interact with Q, even though both can interact with environment.
  - Alphabets of P and Q concealed (only) to each other.
- Repeated general choice between advancing either P or Q:
  - If $P = (v\colon A \to P'(v))$ and $Q = (w\colon B \to Q'(w))$, then $P \interleave Q = (v\colon A \to (P'(v) \interleave Q)) \mathbin{\Box} (w\colon B \to (P \interleave Q'(w)))$
- **Ex:** $(coke \to STOP) \interleave (sprite \to STOP) = (coke \to sprite \to STOP \mid sprite \to coke \to STOP)$.
  - Whereas $(coke \to STOP) \parallel (sprite \to STOP) = STOP$.
- **Ex:** $(coin \to coke \to STOP) \interleave (coin \to STOP) =$
  $coin \to ((coke \to coin \to STOP \mid coin \to coke \to STOP) \sqcap$
  $(coin \to coke \to STOP))$

More laws about interleaving

- $traces(P \mathbin{|||} Q) =$
  $\{s \mid \exists t \in traces(P), u \in traces(Q) \,.\, s \textbf{ interleaves } (t, u)\}$
  - Cf. $traces(P \parallel Q) = traces(P) \cap traces(Q)$, when $\alpha P = \alpha Q$.
- $refusals(P \mathbin{|||} Q) = refusals(P \mathbin{\Box} Q) = refusals(P) \cap refusals(Q)$.
  - Cf. $refusals(P \parallel Q) = \{A \cup B \mid A \in refusals(P), B \in refusals(Q)\}$
- $(P \mathbin{|||} Q) \,/\, s = \bigsqcap_{(t,u) \in S} (P \,/\, t) \mathbin{|||} (Q \,/\, u)$, where
  $S = \{(t, u) \mid t \in traces(P), u \in traces(Q), s \textbf{ interleaves } (t, u)\}$.
  - Note that for any s, S is a *finite* set, so the indexed $\bigsqcap$ can be expressed in terms of binary $\sqcap$.
  - Cf. $(P \parallel Q) \,/\, s = (P \,/\, s) \parallel (Q \,/\, s)$, when $\alpha P = \alpha Q$.

Next time: Communication

- Lecture next Friday (Dec. 13) will be **9:15–11:00**.
- All downhill from now. Will just introduce some streamlined notation and conventions for writing CSP programs, but no truly new concepts.
- Read Chapter 4 of CSP book.
- **Reminder:** HW1 handed out last Tuesday, due December 15.
  - If you haven't started yet, get going!
  - Any questions about the theory part?
  - And once more: whenever you write "$(x \to P) \mid (y \to Q)$", you radiate incompetence...