Non-determinism

**Non-determinism by design**

Dijkstra’s guarded commands (non-deterministic case statements):

\[
\begin{align*}
\text{if } x \leq y & \rightarrow m := x \\
\text{if } x > y & \rightarrow m := y
\end{align*}
\]

- Selection is non-deterministic for \(x = y\).
- The programmer needs to design the alternatives as ‘parallel’ options: G (AB) translate the source code into a concurrent implementation. All true case statements in any language are potentially concurrent and non-deterministic. Numerical non-determinism in concurrent statements (Chap1):

\[
\begin{align*}
\text{arithmetic} & \rightarrow \text{reduce} [1..1000000] \: i \, \text{**} 2.0)
\end{align*}
\]

- Results may be non-deterministic depending on numeric type.
- The programmer needs to understand the numerical implications of out-of-order expressions.

**Non-determinism by interaction**

Selective waiting in Occam2

\[
\text{ALT}
\]

- Guards are referring to boolean expressions and/or channel input operations.
- The boolean expressions are local expressions, i.e. if none of them evaluates to true at the time of the evaluation of the ALT-statement, then the process is stopped.
- If all triggered channel input operations evaluate to false, the process is suspended until further activity on one of the named channels.
- Any Occam2 process can be employed in the ALT-statement.
- The ALT-statement is non-deterministic (there is also a deterministic version: PRI ALT).
Selective Synchronization

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If all conditions are 'true', identical to the previous form.
• If some condition evaluates to 'true' or the accept statement after those conditions is executed as in the preceding form.
• If some condition evaluates to 'false' or Program_Error is raised. Hence it is important that the set of conditions covers all possible states.

This form is identical to Dijkstra's guarded commands.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If all currently open entries have no waiting calls or all entries are closed.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If none of the open entries have waiting calls and none of them can ever be called again. The else alternative is chosen, i.e. the task is terminated.

This situation occurs if ...

• ...all tasks which can possibly call an entry are closed.
• ...or all remaining tasks which can possibly call any of the open entries are waiting on select-terminate statements themselves and none of their open entries can be called either. In this case all these waiting-for-termination tasks are terminated as well.

Selective Synchronization

Message-based selective synchronization in Ada

select accept ::= select (guard) select_alternative |
          | or (guard) select_alternative |
          | else sequence_of_statements |
          |
end select;

guard ::= when <condition> => select_alternative |
          | delay_alternative |
          | terminate_alternative |
          |
select_alternative ::= accept_alternative |
          | sequence_of_statements |
          |
delay_statement ::= delay until delay_expression;

• If none of the entries have waiting calls the process is suspended until a call arrives.
• Exactly one of the entries has waiting calls or this entry is selected.
• If multiple entries have waiting calls or none of these is selected (non-deterministically). The selection can be prioritised by means of the realtime-systems annex.

The code following the select-accept _

select accept _
or accept _
or accept _
end select;

• If none of the entries have waiting calls or the process is suspended until a call arrives.

Basic forms of selective synchronization
(select-guarded-accept)

timed_entry_call | asynchronous_select

• If all currently open entries have no waiting calls or all entries are closed.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.

Message-based selective synchronization in Ada

select accept ::= select (guard) select_alternative |
          | or (guard) select_alternative |
          | else sequence_of_statements |
          |
end select;

guard ::= when <condition> => select_alternative |
          | delay_alternative |
          | terminate_alternative |
          |
select_alternative ::= accept_alternative |
          | sequence_of_statements |
          |
delay_statement ::= delay until delay_expression;

• If none of the entries have waiting calls the process is suspended until a call arrives.
• Exactly one of the entries has waiting calls or this entry is selected.
• If multiple entries have waiting calls or none of these is selected (non-deterministically). The selection can be prioritised by means of the realtime-systems annex.

The code following the select-accept _

select accept _
or accept _
or accept _
end select;

• If none of the entries have waiting calls or the process is suspended until a call arrives.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If all conditions are 'true', identical to the previous form.
• If some condition evaluates to 'true' the accept statement after those conditions is executed as in the previous form.
• If all conditions evaluate to 'false' or Program_Error is raised. Hence it is important that the set of conditions covers all possible states.

This form is identical to Dijkstra's guarded commands.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If current open entries have no waiting calls or all entries are closed.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If none of the open entries have waiting calls and none of them can ever be called again.
• The else alternative is chosen, i.e. the task is terminated.

This situation occurs if ...

• ...all tasks which can possibly call an entry are closed.
• ...or all remaining tasks which can possibly call any of the open entries are waiting on select-terminate statements themselves and none of their open entries can be called either. In this case all these waiting-for-termination tasks are terminated as well.

Selective Synchronization

Message-based selective synchronization in Ada

select accept ::= select (guard) select_alternative |
          | or (guard) select_alternative |
          | else sequence_of_statements |
          |
end select;

guard ::= when <condition> => select_alternative |
          | delay_alternative |
          | terminate_alternative |
          |
select_alternative ::= accept_alternative |
          | sequence_of_statements |
          |
delay_statement ::= delay until delay_expression;

• If none of the entries have waiting calls the process is suspended until a call arrives.
• Exactly one of the entries has waiting calls or this entry is selected.
• If multiple entries have waiting calls or none of these is selected (non-deterministically). The selection can be prioritised by means of the realtime-systems annex.

The code following the select-accept _

select accept _
or accept _
or accept _
end select;

• If none of the entries have waiting calls or the process is suspended until a call arrives.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If some of the open entries have waiting calls and none of them can ever be called again.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If none of the open entries have waiting calls and none of them can ever be called again.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If some of the open entries have waiting calls and none of them can ever be called again.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If none of the open entries have waiting calls and none of them can ever be called again.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If none of the open entries have waiting calls and none of them can ever be called again.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If none of the open entries have waiting calls and none of them can ever be called again.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If none of the open entries have waiting calls and none of them can ever be called again.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If none of the open entries have waiting calls and none of them can ever be called again.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If none of the open entries have waiting calls and none of them can ever be called again.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If none of the open entries have waiting calls and none of them can ever be called again.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If none of the open entries have waiting calls and none of them can ever be called again.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.

Basic forms of selective synchronization
(select-guarded-accept)

select when condition => accept _
or when condition => accept _
or when condition => accept _
end select;

• If none of the open entries have waiting calls and none of them can ever be called again.
• The else alternative is chosen, the associated statements executed and the select statement completes.
• Otherwise as one of the open entries with calling tasks is chosen as above.

This form never suspends the task.
Non-determinism

Selective Synchronization
Timed entry-calls

timed_entry_call ::= select
   entry_call_statement
   (sequence_of_statements)
   delay alternative
   end select;

- If the call is not accepted before the deadline specified by the delay alternative
  or the delay alternative is chosen.

Example
select
   (controller_Request (Seq_Item));
   delay 45.0; delay 45.0;
   try something else
end select;

Non-determinism

Selective Synchronization
Message-based selective synchronization in Ada

Forms of selective waiting

select_statement ::= selective_accept
   | conditional_entry_call
   | timed_entry_call
   | asynchronous_select
   | "underlying concept: Djikstra's guarded commands"

asynchronous_select implemen...t... due to (just a few):

- Operating systems / runtime environments:
  - System load will have an influence on concurrent execution.
  - Message passing systems react load dependent.
  - Networks & communication systems:
    - Traffic will arrive in an unpredictable way (non-deterministic).
    - Example: call after some specified time-out.
  - Computing hardware:
    - Timers drift and clocks have granularities.
    - Processors have out-of-order units.

- Physical systems
- Computer systems

Non-determinism

Correctness of non-deterministic programs

Partial correctness:

\[ P(I) \land \text{terminates}(\text{Program}(I,O)) \Rightarrow Q(I,O) \]

Total correctness:

\[ P(I) \Rightarrow (\text{terminates}(\text{Program}(I,O)) \land Q(I,O)) \]

Safety properties:

\[ (P(I) \land \text{Processes}(I,S)) \Rightarrow Q(S) \]

where \( Q \) means that \( Q \) does always hold

Liveness properties:

\[ (P(I) \land \text{Processes}(I,S)) \Rightarrow \exists Q(S) \]

where \( Q \) means that \( Q \) does eventually hold (and will then stay true) and \( S \) is the current state of the concurrent system

Summary

- Non-determinism by design:
  - Benefits & considerations
    - Non-determinism by interaction:
      - Selective synchronization
      - Selective accepts
      - Selective calls

Correctness of non-deterministic programs:

- Sources of non-determinism
- Predicates & invariants