Australian National University

C05

Non-Determinism

Assignment Project Exam Help

https://powcoder.com

Add WeChat powcoder

Non-determinism by design

Non-determinism by interaction

Non-Determinism

- Non-determinism by *design*:
  A property of a computation which may have more than one result.

- Non-determinism by *interaction*:
  A property of the operation environment which may lead to different sequences of (concurrent) stimuli.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Non-Determinism by Design: Guarded Commands

- Dijkstra's Guarded Command Language (non-deterministic selection):

```
if x ≤ y → m := x
 | x ≥ y → m := y
fi
```

Assignment Project Exam Help
https://powcoder.com

- Result is nondeterministic for `x=y`

Add WeChat powcoder

- The programmer must design alternatives as 'parallel' options: all cases must be covered and overlapping conditions have same result
- All true case statements in any language are potentially concurrent and non-deterministic

E. Dijkstra *EWD472: Guarded commands, non-determinacy and formal derivation of programs*

Non-Determinism by Design: Parallel Reduction

- Numerical non-determinism in concurrent statements (Chapel):

Assignment Project Exam Help

```
writeln (* reduce [i in 1..10] exp (i));
writeln (+ reduce [i in 1..10000000] 1 / (i ** 2.0));
```

https://powcoder.com

Add WeChat powcoder

- Is reduction operation:
  - Commutative? $x \blacksquare y = y \blacksquare x$
  - Associative? $(x \blacksquare y) \blacksquare z = x \blacksquare (y \blacksquare z)$
- Programmers need to understand the numerical implications of out-of-order expressions

Non-Determinism by Design: Motivation

By explicitly leaving the sequence of evaluation or execution undetermined:

Assignment Project Exam Help

- Compiler / runtime environment can directly (i.e. without any analysis) translate source code into a concurrent implementation.

https://powcoder.com

- Implementation may gain significantly in performance
- Programmer does not need to handle details of concurrent implementation (access locks, messages, synchronizations, …)

Add WeChat powcoder

A programming language which allows for those formulations is required!

Current language support: Ada, Chapel, X10, Fortran, Haskell, OCaml, …

Non-Determinism by Interaction (occam)

Selective waiting in occam2:

```
ALT
  Guard1
    Process1
  Guard2
    Process2
```

- Guards refer to boolean expressions and/or channel input operations

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Boolean expressions are local expressions, if none evaluates to true, the process is stopped
- If all triggered channel input operations evaluate to false, the process is suspended pending input on one of the named channels
- Any occam2 process can be employed in the ALT-statement
- Deterministic version: PRI ALT

Non-Determinism by Interaction (occam)

```
ALT
  NumberInBuffer < Size & Append ? Buffer [Top]
    SEQ
      NumberInBuffer := NumberInBuffer + 1
      Top := (Top + 1) REM Size
  NumberInBuffer > 0 & Request ? ANY
    SEQ
      Take ! Buffer [Base]
      NumberInBuffer := NumberInBuffer - 1
      Base := (Base + 1) REM Size
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

See also Communicating Sequential Processes (Hoare 1978)

Synchronization on input-channels only:
to initiate the sending of data (`Take ! Buffer [Base]`),
a request must be made which triggers the condition: (`Request ? ANY`)

Non-Determinism by Interaction (POSIX)

```
int pselect(int n, fd_set *readfds, fd_set *writefds, fd_set *exceptfds,
            const struct timespec *ptimeout, sigset_t *sigmask);
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

with:

- `n` = maximum of any file descriptor in any of the sets, plus one
- on return, `fd_set`s are reduced to the channels which were triggered

Implements some features of general selective waiting:

- returns if one or more I/O channels have been triggered or error occurred
- branching into individual code sections is not provided
- guards are not provided; after return, must test each channel in the read/write/exception sets

Message-Based Selective Synchronization in Ada

Forms of selective waiting:

```
select_statement ::= selective_accept
                   | conditional_entry_call
                   | timed_entry_call
                   | asynchronous_select
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

underlying concept: Dijkstra's guarded commands

`selective_accept` implements

- wait for multiple rendezvous at any one time
- time-out if no rendezvous is forthcoming within a specified time
- withdraw offer to communicate if no rendezvous available immediately
- terminate if no clients can possibly call its entries

Ada Reference Manual, Selective Accept

Basic Forms of Selective Synchronization

(select-accept)

```
select
   accept …
or
   accept …
or
   accept …
…
end select;
```

- If none of the entries have waiting calls: the process is suspended until a call arrives.
- If exactly one of the entries has waiting calls: this entry is selected.
- If multiple entries have waiting calls: one of those is selected (non-deterministically). The selection can be prioritized by means of the real-time-systems annex
- The code following the selected entry (if any) is executed and the select statement completes.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Basic Forms of Selective Synchronization

(select-guarded-accept)

```
select
 when <condition1> => accept …
or
 when <condition2> => accept …
or
 when <condition3> => accept …
…
end select;
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- If all conditions are ‘true’: identical to the previous form
- If some conditions evaluate to ‘true’: the guarded accept statements are treated as per select-accept
- If all conditions evaluate to ‘false’: Program_Error is raised..
  Hence it is important that the set of conditions covers all possible states
- Identical to Dijkstra’s guarded commands

Basic Forms of Selective Synchronization

(select-guarded-accept-else)

```
 select
  when <condition1> => accept …
or
  when <condition2> => accept …
or
  when <condition3> => accept …
…
else
   <statements>
end select;
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- If all currently open entries have no waiting calls or all entries are closed: the `else` alternative is executed and the select statement completes
- Otherwise: one of the open entries with waiting calls is chosen as per select-guarded-accept.
- Does not suspend the task
- Enables a task to *withdraw* its offer to communicate if no task is currently waiting

Basic Forms of Selective Synchronization

(select-guarded-accept-delay)

```
select
 when <condition1> => accept …
or
 when <condition2> => accept …
…
or
  when <condition> => delay [until] …
    <statements>
or
  when <condition> => delay [until] …
…
end select;
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- If no open entries have waiting calls before deadline specified by earliest open delay alternative, the earliest delay alternative is executed
- Otherwise:
  one of the open entries with waiting calls is chosen as above
- Enables a task to withdraw its offer to communicate if no other task is calling after some time

Basic Forms of Selective Synchronization

(select-guarded-accept-terminate)

```
select
 when <condition1> => accept …
or
 when <condition2> => accept …
…
or
  when <condition> => terminate;
end select;
```

terminate cannot be mixed with **else** or **delay**

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- If none of the open entries have waiting calls and none of them can ever be called again, the task is terminated

This situation occurs if:

- … all tasks which can possibly call any open entries have terminated
- or … all remaining tasks which can possibly call on any of the open entries are waiting themselves

Message-Based Selective Synchronization in Ada

Forms of selective waiting:

```
select_statement ::= selective_accept
                   | conditional_entry_call
                   | timed_entry_call
                   | asynchronous_select
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

underlying concept: Dijkstra's guarded commands

`conditional_entry_call` and `timed_entry_call` implement the possibility to withdraw an outgoing call (might be restricted if calls have already been partly processed)

Ada Reference Manual, Conditional Entry Call
Ada Reference Manual, Timed Entry Call

Conditional Entry Calls

```
conditional_entry_call ::=
  select
    entry_call_statement
    [sequence_of_statements]
  else
    sequence_of_statements
  end select;
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- If call is not accepted immediately, the `else` alternative is chosen
- Useful e.g. to probe the state of a server before committing to a potentially blocking call
- Although it is tempting to use this statement for “busy-waiting”, better alternatives are available
- Only one entry call and one else alternative

Timed Entry Calls

```
timed_entry_call ::=
  select
    entry_call_statement
    [sequence_of_statements]
  or
    delay_alternative
  end select;
```

- If the call is not accepted before the deadline specified by the delay alternative, the `delay` alternative is chosen
- Useful to withdraw an entry call after some specified time-out
- Only one entry-call and one delay alternative

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Message-Based Selective Synchronization in Ada

Forms of selective waiting:

```
select_statement ::= selective_accept
                   | conditional_entry_call
                   | timed_entry_call
                   | asynchronous_select
```

Assignment Project Exam Help
https://powcoder.com

underlying concept: Dijkstra's guarded commands

Add WeChat powcoder

`asynchronous_select` implements the possibility to escape a running code block due to an event from outside this task. (Outside the scope of this course - see Real-Time Systems)

Sources of Non-Determinism

As concurrent entities are not in “lockstep” synchronization, they arrive at synchronization points in non-deterministic order due to e.g.:

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

- Operating systems / runtime environments:
  - Schedulers
  - Message passing systems
- Networks & communication systems:
  - Multiple routing paths
  - Communication systems congestion
- Computing hardware:
  - Timer drift and clocks granularity
  - Out-of-order execution
- … computer systems connected to the physical world are intrinsically non-deterministic.

Correctness of Non-Deterministic Programs

- Partial correctness:
  - (P(I) ∧ terminates (Program(I,O))) → Q(I,O)
- Total correctness:
  - P(I) → (terminates (Program(I,O)) ∧ Q(I,O))
- (P(I) ∧ Processes(I,S)) → □Q(I,S)

  where □Q means that Q *always* holds and S is the current state of the concurrent system
- (P(I) ∧ Processes(I,S)) → ◇Q(I,S)

  where ◇Q means that Q *eventually* holds

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Correctness of Non-Deterministic Programs

- Correctness predicates need to hold true
  irrespective of the actual sequence of interaction points
  or
  for all possible sequences of interaction points.
- Therefore correctness predicates need to be based on *invariants*, i.e. invariant predicates which are independent of the potential execution sequences, yet support the overall correctness predicates.

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Correctness of Non-Deterministic Programs

- For example: “Mutual exclusion accessing a specific resource holds true, for all possible numbers, sequences or interleavings of requests to it”
- E.g. invariant: the number of writing tasks inside a protected object is less or equal to one.
- Such invariants are the only practical way to guarantee correctness, as enumerating all possible cases and proving them individually is in general not feasible

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Correctness of Non-Deterministic Programs

```
select
 when <condition1> => accept …
or
 when <condition2> => accept …
or
 when <condition3> => accept …
…
end select;
```

Assignment Project Exam Help
https://powcoder.com
Add WeChat powcoder

Concretely:

- Whenever you use a non-deterministic statement like the one on the left you need to formulate an invariant which holds regardless of which alternative is actually chosen
- Similar to finding loop invariants in sequential programs