Logical Correctness The Logical Behavioral Semantics

Causality issues The logical coherence law Logical reactivity and determinism

## Synchronous Languages—Lecture 05

Prof. Dr. Reinhard von Hanxleden

Christian-Albrechts Universität Kiel Department of Computer Science Real-Time Systems and Embedded Systems Group

11 Nov 2016 Last compiled: November 10, 2016, 13:54 hrs

Lecture 05



Synchronous Languages

Logical Correctness

1. How do concurrent threads in Esterel communicate?

2. What is the difference between *weak* and *strong* abortion?

The Logical Behavioral Semantics

The 5-Minute Review Session

### Logical Correctness

Overview

Causality issues The logical coherence law Logical reactivity and determinism Instantaneous Feedback

Logical Correctness The Logical Behavioral Sem

Synchronous Languages

Causality issues Logical reactivity and determinism

Lecture 05

# **Causality Problems**

CAU

present A else emit A end

abort	
pause;	
emit A	
when A	

#### present A then pause end; emit A

#### It's easy to write contradictory programs

Unfortunate side-effect of instantaneous communication coupled with the single valued signal rule

These sorts of programs are erroneous and flagged by the Esterel compiler as incorrect

► *Note:* the first and third example are considered valid in SCEst, see later ....

- 3. What is the difference between *aborts* and *traps*? 4. What is *syntactic sugar*, and what is it good for?
- 5. What is the multiform notion of time?

Slide 1

Slide 3

**Causality issues Causality issues** Logical Correctness The logical coherence law Logical Correctness The logical coherence law The Logical Behavioral Sem The Logical Behavioral Semantics Logical reactivity and determinism Logical reactivity and determinism **Causality Problems** Causality Example Red statements reachable emit A; present B then emit C end; Г present A else emit B end; abort emit A Analysis done by original compiler: when immediate B Can be very complicated ٦ ▶ After emit A runs, there's a static path to emit B because of instantaneous communication ► Therefore, the value of B cannot be decided yet Г present A Execution procedure deadlocks: Program is bad then emit B end; ] CAU Slide 5 CAU Slide 7 Synchronous Languages Lecture 05 Synchronous Languages Lecture 05 Causality issues Causality issues Logical Correctness Logical Correctness The logical coherence law The Logical Behavioral Semantics The Logical Behavioral Sema Logical reactivity and determinism Logical reactivity and determinism Causality Example Causality Red statements reachable emit A; present B then emit C end; present A else emit B end;

- Definition has evolved since first version of the language
- Original compiler had concept of "potentials"
  - Static concept: at a particular program point, which signals could be emitted along any path from that point
- Current definition based on "constructive causality"
  - Dynamic concept: whether there's a "guess-free proof" that concludes a signal is absent

#### Analysis done by later compilers:

- After emit A runs, it is clear that B cannot be emitted because A's presence runs the "then" branch of the second present
- B declared absent, both present statements run
- Program is OK

causality issues ectness
The logical coherence law Logical reactivity and determinism Instantaneous Exactly activity

## Logical Correctness

- ► The intuitive semantics:
  - Specifies what should happen when executing a program
- However, also want to guarantee that
  - Execution actually exists (at *least* one possible execution)
  - Execution is unique (at most one possible execution)
- Need extra criteria for this!
- > The apparently simplest possible criterion: logical correctness

Lecture 05

Slide 0

#### Logical Correctness

#### Given:

- Program P and input event I
- *P* is logically reactive w.r.t. *I*:
  - ▶ There is at least one logically coherent global status
- *P* is logically deterministic w.r.t. *I*:
  - There is at most one logically coherent global status
- *P* is logically correct w.r.t. *I*:
- ► *P* is both logically reactive and deterministic

*P* is logically correct:

- ▶ *P* is logically correct w.r.t. *all* possible input events
- Is logical correctness decidable?
  - Yes!

C A U Synchronous Languages	Lecture 05	Slide 11
-----------------------------	------------	----------

Logical Correctness
The Logical Behavioral Semantics
The Logical Participation of the logical coherence law
Logical reactivity and deter
Instantaneous Feedback

Synchronous Languages

## Logical Correctness

CAU

Recall:

- ► Signal S is absent by default
- ▶ Signal S is present if an emit S statement is executed

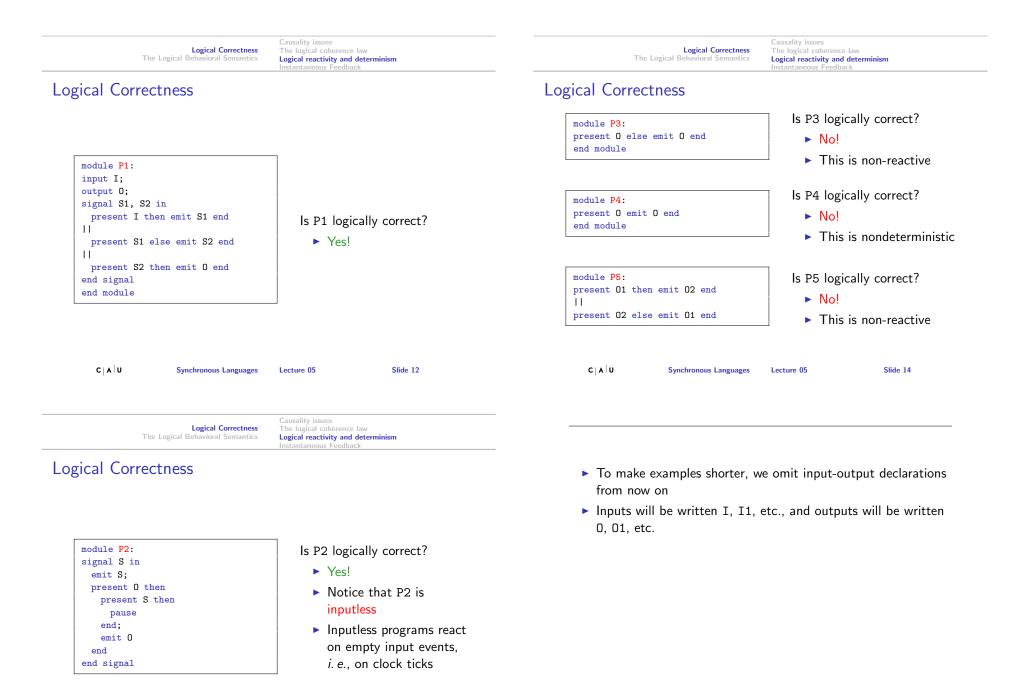
The Logical Coherence Law:

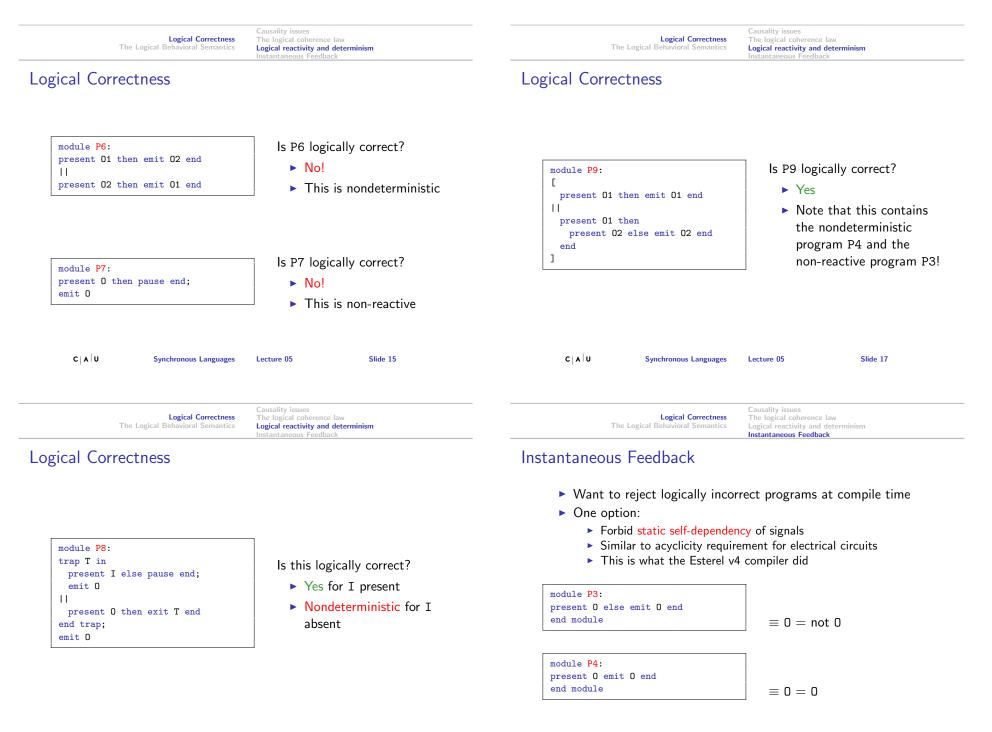
A signal S is present in a tick if and only if an emit S statement is executed in this tick.

#### Logical Correctness requires:

 There exists exactly one status for each signal that respects the coherence law

- Pure Esterel programs can be analyzed for logical correctness by performing exhaustive case analysis
- Given the status of each input signal, one can make all possible assumptions about the global status and check them individually
- Therefore, logical correctness is decidable
- We here generally consider just a single reaction. However, in general one also has to consider all possible sequences of reactions and all possible program states. As there is a finite number of program states, this is still decidable.





Lecture 05

Causality issues Logical Correctness The logical coherence law Logical Correctness The logical coherence law The Logical Behavioral Semantics The Logical Behavioral Semantics Logical reactivity and determinisn Logical reactivity and determinisn Instantaneous Feedback Instantaneous Feedback Instantaneous Feedback Logical Correctness—Assessment We now introduced logical correctness But do we want to use it as basis for the language? However, forbidding cycles would also reject the following: ☺ sound © sometimes unintuitive (consider P9) module GoodCycle1: © computationally complex present I then Alternative 1: allow only programs that are acyclic present 01 then emit 02 end ③ simple else  $\odot$  too restrictive (consider GoodCycle1/2) present 02 then emit 01 end Alternative 2: accept everything for which the compiler finds a end present static execution schedule © relatively simple for the compiler © definition not precise, depends on abilities of compiler ▶ 01 and 02 cyclically depend on each other (different compilers accept different programs) ► However, any given status of I breaks the cycle Alternative 3: the constructive semantics. ② analysis not trivial © clear semantics CAU Slide 10 CAU Synchronous Languages Slide 21 Synchronous Languages Lecture 05 Lecture 05 Notation and Definitions Logical Correctness Logical Correctness The logical coherence law The Basic Broadcasting Calculus The Logical Behavioral Semantics The Logical Behavioral Semantics Logical reactivity and determinisn Instantaneous Feedback Reactivity and Determinism Instantaneous Feedback Overview

### Overview

#### Logical Correctness

## The Logical Behavioral Semantics

Notation and Definitions The Basic Broadcasting Calculus Transition Rules Reactivity and Determinism

#### ► Here the cycle is neutralized with a delay

module GoodCycle2:

pause;

present 01 then emit 02 end;

present 02 then emit 01 end

In general, requiring acyclicity turns out to be inadequate to Esterel practice

Slide 22

Notation and Definitions The Basic Broadcasting Calculus Transition Rules Reactivity and Determinism

#### The Semantics of Esterel

- 1. Logical Behavioral Semantics
  - Rewriting rules defining reactivity, determinism, and logical correctness
  - Signal coherence law embedded in rules for local signals
- 2. Constructive Behavioral Semantics
  - Refines logical behavioral semantics
  - Based on *must* and *cannot* analysis
- 3. Logical/Constructive State Behavioral Semantics
  - Replaces rewriting with marking of active delays (v5 debugger)
- 4. Constructive State Operational Semantics
  - Defines reaction as sequence of microsteps (v3 compiler)
- 5. Constructive Circuit Semantics
  - Translates Esterel programs into Boolean digital circuits (v5 compiler)

С	AU	Synchronous Languages	Lecture 05	Slide 23

#### Notation and Definitions

- Sort S: A set of signals
- Signal statuses:  $B = \{+,-\}$
- Event E:
  - Given sort S, defines status  $E(s) \in B$  for each  $s \in S$
  - Obtain sort of E as S(E) = S
- ► Two equivalent representations for *E*:
  - ► As subset of S:  $E = \{s \in S \mid E(s) = +\}$
  - As a mapping from S to B:  $E = \{(s, b) \mid b = E(s)\}$

CIAU	Synchronous Languages	Lecture 05	Slide 24
------	-----------------------	------------	----------

- The logical behavioral semantics accepts more programs than we would like (for example, program P9 presented in Lecture 03)
- However, the logical behavioral semantics is important in that all other semantics should be a refinement of it, and it is also a natural starting point
- The constructive semantics are equivalent; the constructive behavioral semantics is the most intuitive, and can be derived fairly directly from the logical behavioral semantics, so we will focus on these two semantics here
- Note that the terminology (and categorization) used in different references (and sometimes within the same reference—e.g., in Berry's draft book) is a bit in flux; the keywords to look out for to distinguish which is which are "logical" vs. "constructive", "state" and "behavioral" vs. "operational"
- ▶ In this class, will focus on semantics 1, 2, and 5

 Allowing to represent events in alternate ways somewhat simplifies the subsequent presentation of the rewriting rules

#### Notation and Definitions

- ▶ Write  $s^+ \in E$  iff E(s) = +
- Write  $s^- \in E$  iff E(s) = -
- Write  $E' \subset E$  iff  $\forall s \in S(E') : s^+ \in E' \Longrightarrow s^+ \in E$
- Given signal s, define singleton event  $\{s^+\}$ :

Logical Correctness

The Logical Behavioral Semantics

- $\{s+\}(s) = +$   $\forall s' \neq s : \{s+\}(s') = -$
- Given signal set S and signal  $s \in S$ , write  $S \setminus s = S \{s\}$

Notation and Definitions

The Basic Broadcasting Calculus

- Given *E* and  $s \in S(E)$ , write  $E \setminus s$  to denote event of sort  $S(E) \setminus s$ , which coincides with E on all signals but s
- ▶ Define  $E * s^b$  as event E' of sort  $S(E) \cup \{s\}$  with • E'(s) = b, E'(s') = E(s') for  $s' \neq s$

Notation	and	Definitions
----------	-----	-------------

- ▶ Will present formal semantics as Plotkin's Structural Operational Semantics (SOS) inference rules
- ► Behavioral Semantics formalizes reaction of program *P* as behavioral transition

$$P \xrightarrow{O}{I} P'$$

- ► *I*: input event
- ► *O*: output event
- ► *P*': derivative of *P*—the program for the next instance

C A U	Synchronous Languages	Lecture 05	Slide 25	C A U	Synchronous Languages	Lecture 05	Slide 26

• Note that in the definition of  $E * s^b$ , s may or may not be in S(E); in the former case, the status of s in E is lost in  $E * s^b$ 

### Notation and Definitions

Auxiliary statement transition relation:

The Logical Behavioral Semantics

Logical Correctness



Notation and Definitions

The Basic Broadcasting Calculus

- ▶ *p*: program body (of *P*)
- E: event defining status of all signals declared in scope of p
- E': event composed of all signals emitted by p in the reaction
- ► *k*: completion code returned by *p* (0 iff *p* terminates)
- ▶ p': derivative of p
- Logical coherence (or broadcasting invariant):



- Notation and Definitions
  - Given:
    - Program P with body p
    - Input event I
  - Define program transition of *P* by statement transition of *p*:

$$P \xrightarrow[l]{O} P'$$
 iff  $p \xrightarrow[l]{O,k} p'$  for some  $k$ 

 These program transitions, yielding an output reaction O and a derivative P', determine the logical behavioral semantics of P

CIAU	Synchronous Languages	Lecture 05	Slide 28
------	-----------------------	------------	----------

- Here, we consider an Esterel program to consist of an input/output signal interface and an executable body
- Note that the event E is an assumption in the sense of the logical semantics

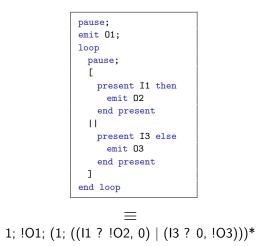
 Note how the definition of the program transition reflects the logical coherence

### The Basic Broadcasting Calculus

- For concise presentation of rules: Replace Esterel syntax with terser process-calculus syntax
- Use parenthesis for grouping statements

	nothing	0
	pause	1
	emit s	<u> </u> s
	present $s$ then $p$ else $q$ end	s?p,q
	p; q	p; q
	loop <i>p</i> end	<i>p</i> *
	$p \parallel q$	pq
	signal s in p end	p \ s
	suspend $p$ when $s$ end	s ⊃ p
	trap $T$ in $p$ end	{ <i>p</i> }
	exit <i>T</i>	k with $k \ge 2$
	[no concrete syntax]	↑p
C A U	Synchronous Languages Lecture	05 Slide 29

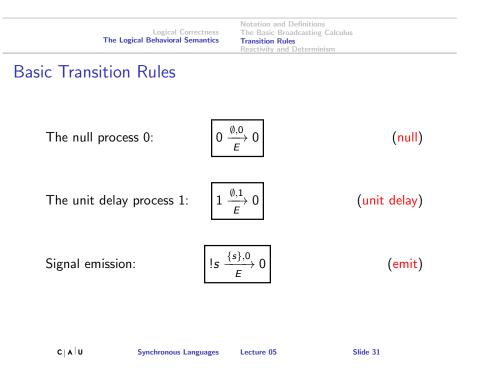
### Example



CAU	Synchronous Languages	Lecture 05	Slide 30

#### Recall: trap T in p end

- Defines a lexically scoped exit point T for p
- Immediately starts its body p and behaves as p until termination or exit
- ► If *p* terminates, so does the trap statement
- If p exits T, then the trap statement terminates instantaneously
- If p exits an enclosing trap U, this exit is propagated by the trap statement
- ► Is part of pure Esterel



### **Deduction Rules**

- In addition to simple transition rules, will also use deduction rules
- ▶ Hypothesis: If sub-instructions behave like this ....

$p_1 \xrightarrow{E_1',k_1} p_1'$	$p_2 \xrightarrow{E'_2,k_2}{E} p'_2$	Other hypotheses
Instruction $(p_1, p_2)$	$\frac{E'(E'_1,E'_2)}{E}$	$\xrightarrow{_{1},k_{2})} Instruction'(p_{1}',p_{2}')$

 Conclusion: ... then the compound instruction behaves like that

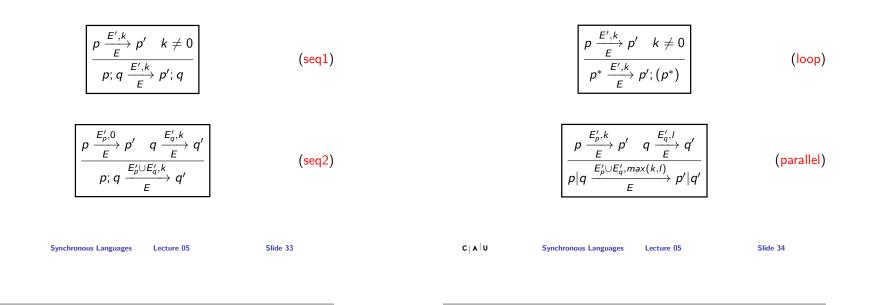
CAU	Synchronous Languages	Lecture 05	Slide 32
-----	-----------------------	------------	----------

- The null process 0 terminates instantaneously and rewrites into itself
- The unit delay process 1 waits in the current reaction and rewrites itself into 0 for the next reaction

Deduction Rules—Looping and Parallel

#### Deduction Rules—Sequencing

CAU



- If the first component of a sequence waits, the sequence also waits
  - For reasons that will become clear later, write waiting as k ≠ 0 instead of k = 1
- If the first component of a sequence terminates, the second is started (in zero delay), in the same environment *E*, and the emitted signals are merged
  - Using same E for both premises implements forward broadcasting from p to q, as broadcasting invariant of first premise implies E'<sub>P</sub> ⊂ E
  - However, with the same reasoning we have backward broadcasting from q to p, conflicting with our requirement for causality—will rule this out later

- Note how the global broadcasting invariant expresses that signals are broadcast between parallel branches: E'<sub>p</sub> ∪ E'<sub>q</sub> ⊂ E holds iff both E'<sub>p</sub> ⊂ E and E'<sub>q</sub> ⊂ E hold
- Note that parallel constructs where all threads have terminated get cleaned up by the (seq2) rule or (trap1)

Deduction Rules—Conditional

$$\frac{s^{+} \in E \quad p \xrightarrow{E',k} p'}{s?p,q \xrightarrow{E',k} p'} \qquad (present +)$$

$$\frac{s^{-} \in E \quad q \xrightarrow{E',k} p'}{s?p,q \xrightarrow{E',k} q'} \qquad (present -)$$

Zero delay: can use decision trees to test for arbitrary Boolean conditions:

•  $(s_1 \wedge s_2)?p, q \text{ is } s_1?(s_2?p, q), q$ 

• 
$$(s_1 \vee s_2)?p, q \text{ is } s_1?p, (s_2?p, q)$$

► ¬s?p, q is s?q, p

CAU	Synchronous Languages	Lecture 05	Slide 35
•	eynemenede Eungauges	Ecolute oo	0.000

Deduction Rules—Restriction

$$\frac{p \xrightarrow{E' * s^+, k} p' \quad S(E') = S(E) \setminus s}{p \setminus s \xrightarrow{E', k} p' \setminus s}$$
(sig +)

Notation and Definitions

Transition Rules Reactivity and De

The Basic Broadcasting Calculus

$$\frac{p \xrightarrow{E' * s^-, k} p' \quad S(E') = S(E) \setminus s}{p \setminus s \xrightarrow{E', k} p' \setminus s}$$
(sig -)

Note: This also properly handles nested restrictions of the same signal

C | A | U Synchronous Languages Lecture 05 Slide 36

#### Example: loop emit S; pause; emit T end.

In the process calculus: (!S; 1; !T)\*

Calculating initial reaction, as a derivative tree (Ableitungsbaum):

$$\frac{\frac{\frac{|S - \{S\}, 0\}}{\{S\}} \circ 0, -1 - \frac{\emptyset, 1}{\{S\}} \circ 0}{\frac{|S; 1 - \{S\}, 1\}}{\{S\}} \circ 0} (\operatorname{seq2})}{\frac{|S; 1; |T - \frac{\{S\}, 1}{\{S\}} \circ 0; |T|}{\{S\}} (\operatorname{seq1})} (\operatorname{loop})}{(!S; 1; !T) * \frac{\{S\}, 1}{\{S\}}} \circ 0; !T; (!S; 1; !T) * (\operatorname{loop})}$$

See next note for an alternative notation. Similarly, for next reaction (and all following):  $0; !T; (!S; 1; !T) * \xrightarrow{\{S,T\},1} 0; !T; (!S; 1; !T) *$   The additional sort condition expresses that the sort of E' does not contain s—this avoids propagating the local status of s outside the p\s statement

Another notation for initial reaction of example from previous note:

$$\begin{array}{cccc} !S \xrightarrow{\{S\},0} & 0, & 1 \xrightarrow{\emptyset,1} & 0 & \stackrel{(\text{seq2})}{\Longrightarrow} & !S; 1 \xrightarrow{\{S\},1} & 0 \\ & \stackrel{(\text{seq1})}{\Longrightarrow} & !S; 1; !T \xrightarrow{\{S\},1} & 0; !T \\ & \stackrel{(\text{loop})}{\Longrightarrow} & (!S; 1; !T) * \xrightarrow{\{S\},1} & 0; !T; (!S; 1; !T) * \end{array}$$

Logical Correctness The Logical Behavioral Semantics

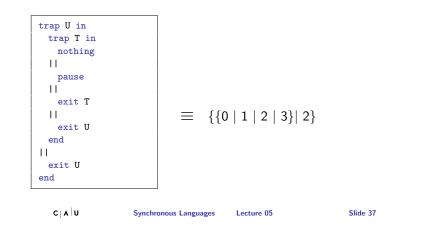
#### Traps—Example

- The trap exit encoding is
  - k = 2 if the closest enclosing trap is exited, and
  - k = n + 2 if *n* trap declarations have to be traversed

Notation and Definitions

Transition Rules

The Basic Broadcasting Calculus



Notation and Definitions The Basic Broadcasting Calculus **Transition Rules** Reactivity and Determinism

## The Shift Operator

- ("shift") shifts exit numbers of p by 1 when placing p in a
   trap block
- May use  $\uparrow$  in definitions of derived operators

suspend $p$ when immediate $s$	s ·⊃ p	$\equiv$ {(s?1,2)*}; s $\supset$ p
await immediate s; p	$s \mapsto p$	$\equiv \{(s?(\uparrow p; 2), 1)^*\}$
await s; p	$s \Rightarrow p$	$\equiv$ 1; s $\cdot \Rightarrow$ p
weak abort $p$ when immediate $s$	$s \cdot > p$	$\equiv \{(\uparrow p; 2) \mid s \cdot \Rightarrow 2\}$
weak abort $p$ when $s$		$\equiv \{(\uparrow p; 2) \mid s \Rightarrow 2\}$
abort $p$ when immediate $s$	$s \cdot \gg p$	$\equiv s \cdot > (s \cdot \supset p)$
abort $p$ when $s$	$s \gg p$	$\equiv s > (s \supset p)$

CA	U	Synchronous Languages	Lecture 05	Slide 39
----	---	-----------------------	------------	----------

Logical Correctness The Logical Behavioral Semantics	Notation and Definitions The Basic Broadcasting Calculus <b>Transition Rules</b> Reactivity and Determinism
---	--

### Two Operators on Completion Codes

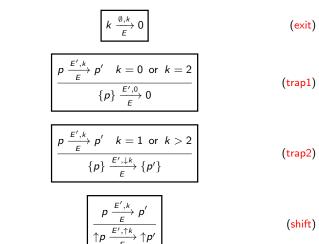
The ↓k operator computes completion code of {p} from that of p:

$$\downarrow k = 0 \qquad \text{if } k = 0 \quad \text{or } k = 2$$
  
$$\downarrow k = 1 \qquad \text{if } k = 1$$
  
$$\downarrow k = k - 1 \quad \text{if } k > 2$$

- The ↑k operator computes completion code of ↑p from that of p; want {↑p} ≡ p
  - $\begin{array}{ll} \uparrow k = k & \quad \text{if } k = 0 \quad \text{or} \quad k = 1 \\ \uparrow k = k + 1 & \quad \text{if } k > 1 \end{array}$



CAU

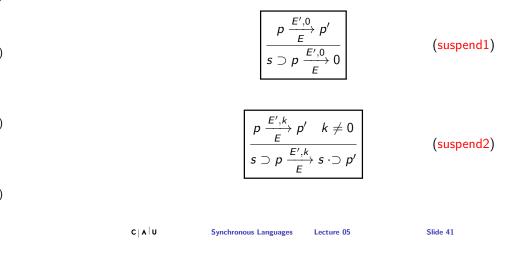


Lecture 05

Synchronous Languages

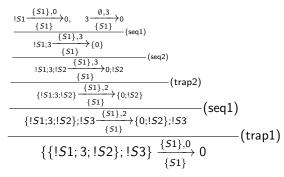
Slide 40

#### Deduction Rules—Suspension



Logical Correctness The Logical Behavioral Semantics Notation and Definitions The Basic Broadcasting Calculus Transition Rules Reactivity and Determinism

Note: It might be a bit surprising that in (trap2), the braces (trap scope) remain in the program derivative when an internal exception is propagated up. However, this works fine: the  $\downarrow k$  operator keeps lowering the trap completion code, and as soon as we reach the trap scope corresponding to the exception, everything reduces to nothing. See for example { { !S1; 3; !S2 }; !S3 }:



## Reactivity and Determinism

- ▶ Definition: Program P is logically reactive (resp. logically deterministic) w.r.t. an input event I if there exists at least (resp. at most) one program transition P → P' for some output event O and program derivative P'
- Definition: Program P is logically correct if it is logically reactive and logically deterministic
- ▶ How about (*s*?!*s*, 0)?
- ▶ And how about (s?0, !s)?

• there exists a unique proof of the unique transition  $P \xrightarrow{O}_{l} P'$ .

## Reactivity and Determinism

non-determinism

event / iff

• Consider  $(s?!s, 0) \setminus s$ 

I/O determinism still leaves room for internal

Forbidden in constructive semantics

Definition: Program P is strongly deterministic for an input

▶ *P* is reactive and deterministic for this event. and

Notation and Definitions The Basic Broadcasting Calculus Transition Rules Reactivity and Determinism

## Summary (2/3)

- P is logically reactive w.r.t. input I if there is at least one logically coherent global status
- P is logically deterministic w.r.t. I if there is at most one logically coherent global status
- P is logically correct w.r.t. I if P is both logically reactive and deterministic
- P is logically correct if P is logically correct w.r.t. all possible input events

C   A   U Synchronous Languages	Lecture 05 Slide 43	C A	U Synchronous Languages	Lecture 05	Slide 45
Logical Correctness The Logical Behavioral Semantics			Logical Correctness Notation and Definitions The Logical Behavioral Semantics Transition Rules Reactivity and Determinism		

# Summary (1/3)

- The intuitive semantics specifies what should happen when executing a program
- However, also want to guarantee that exactly one possible execution exists that satisfies the intuitive semantics
- The Logical Coherence Law specifies that a signal S is present in a tick if and only if an "emit S" statement is executed in this tick
- Logical Correctness requires that there exists exactly one status for each signal that respects the coherence law

# Summary (3/3)

- There exist several semantics for the Esterel language—one important distinction is between *logical* and *constructive* semantics, the latter being a refinement of the former
- We started discussing the logical behavioral semantics, expressed in Plotkin's Structural Operational Semantics, with basic transition rules and deduction rules
- We formally defined reactivity, determinism, logical correctness, and strong determinism

Slide 46

Logical Correctness The Logical Behavioral Semantics

Notation and Definitions The Basic Broadcasting Calculus Transition Rules Reactivity and Determinism

## To Go Further

- Gérard Berry, The Constructive Semantics of Pure Esterel, Draft book, current version 3.0, Dec. 2002
   http://www-sop.inria.fr/members/Gerard.Berry/ Papers/EsterelConstructiveBook.zip
- Gérard Berry, Preemption in Concurrent Systems, In Proceedings FSTTCS 93, Lecture Notes in Computer Science 761, pages 72-93, Springer-Verlag, 1993, http://citeseerx.ist.psu.edu/viewdoc/summary?doi= 10.1.1.42.1557

C A U Synchronous Languages Lecture 05 Slide 47