Distributed, Modular HTL

Thomas A. Henzinger
EPFL / IST Austria

Christoph M. Kirsch
University of Salzburg

Eduardo R. B. Marques
University of Porto

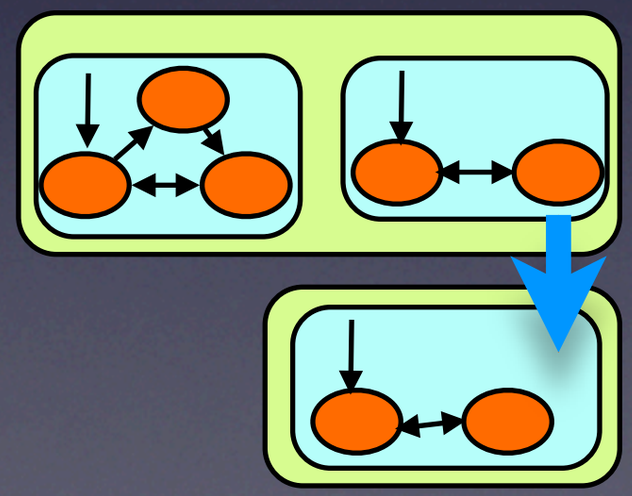
Ana Sokolova
University of Salzburg

RTSS 2009 - 30th IEEE Real-Time Systems Symposium
December 3, 2009

Distributed, Modular HTL

- HTL = Hierarchical Timing Language [EMSOFT 2006]
- Modularity = compositionality

- HTL is modular (syntax and semantics)
- How modular is HTL compilation ?
- How modular is HTL distribution ?



Time-Portable Programming

Giotto

[EMSOFT 2001, Proceedings of the IEEE 2003]

HTL

[EMSOFT 2006, RTSS 2009]

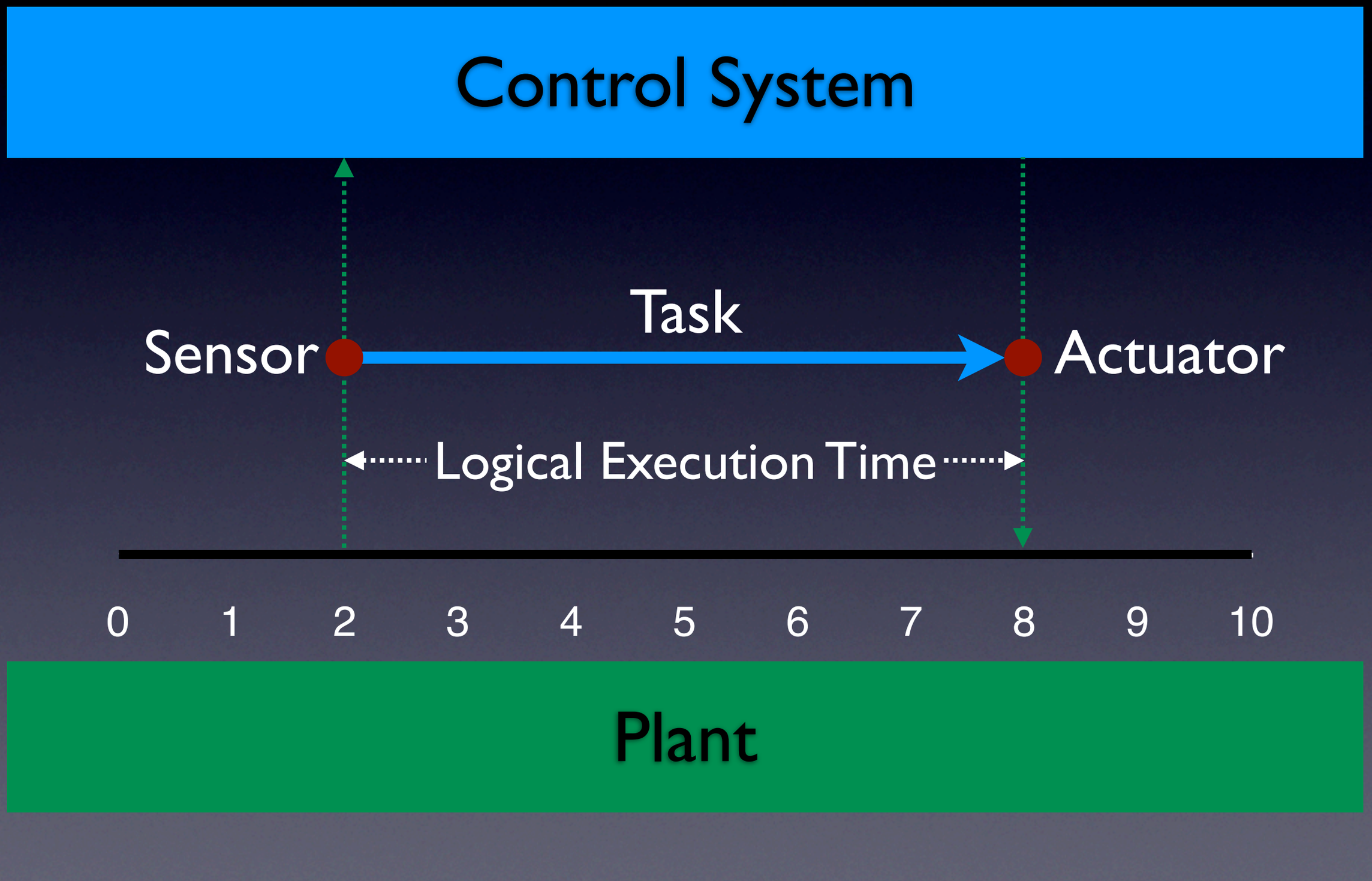
Exotasks

[LCTES 2007, TECS 2009]

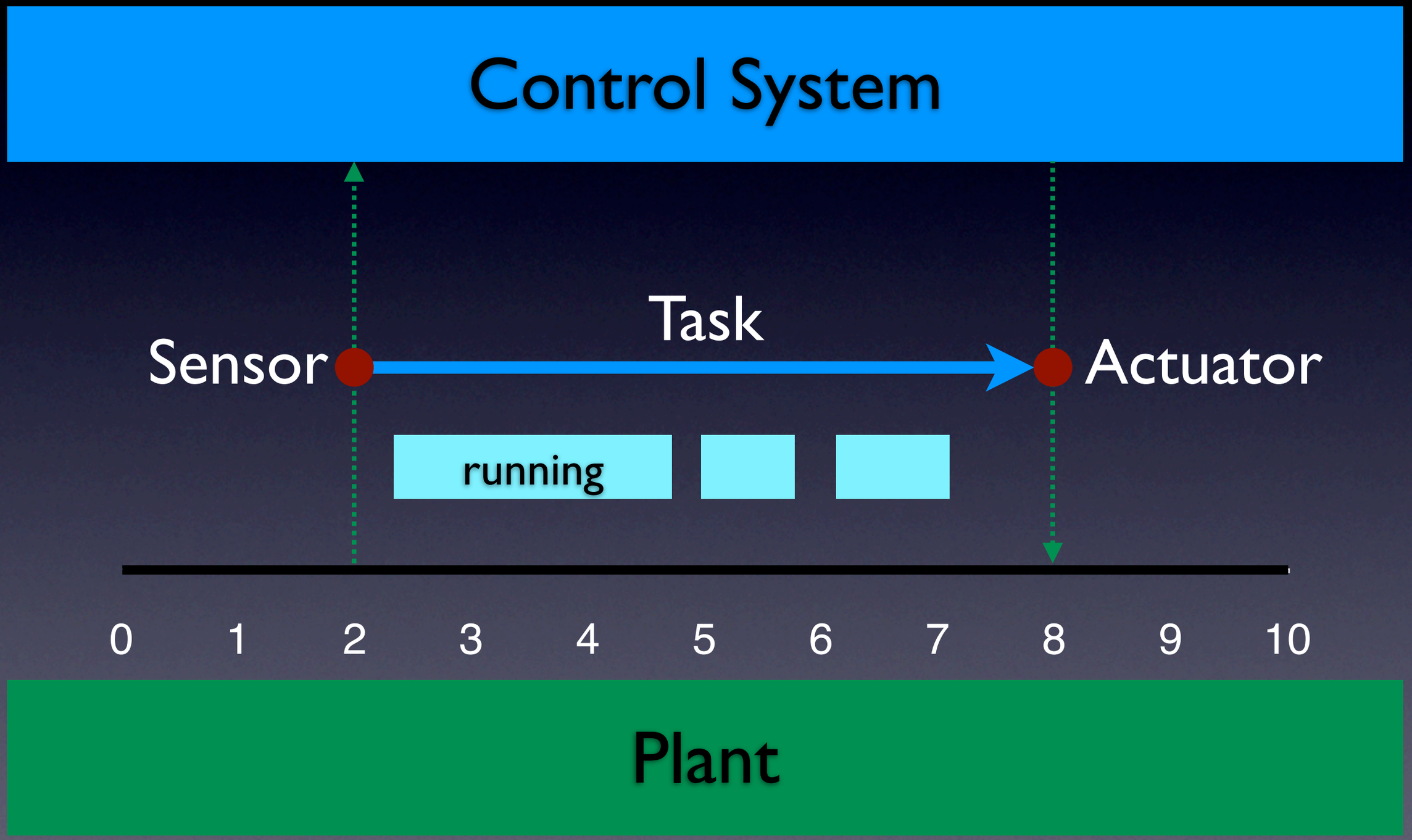
Tiptoe

[USENIX 2008, IIES 2009, SIES 2009]

Logical Execution Time



Actual Execution Time



Time Determinism

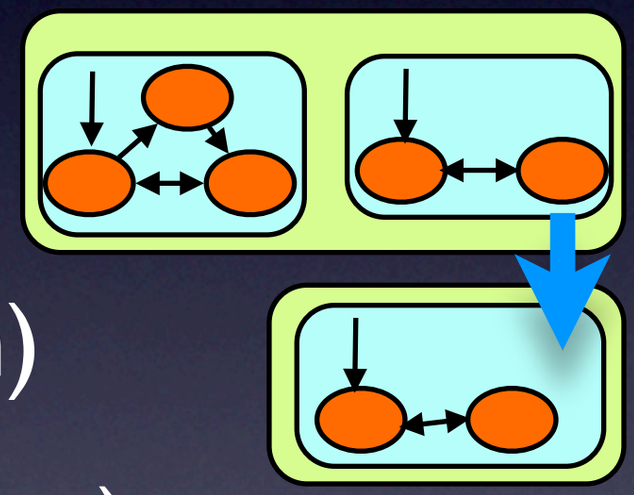
Control System

A system's I/O behavior is time-deterministic if,
for all sequences of input values and times,
the system always produces
unique sequences of output values and times.

Plant

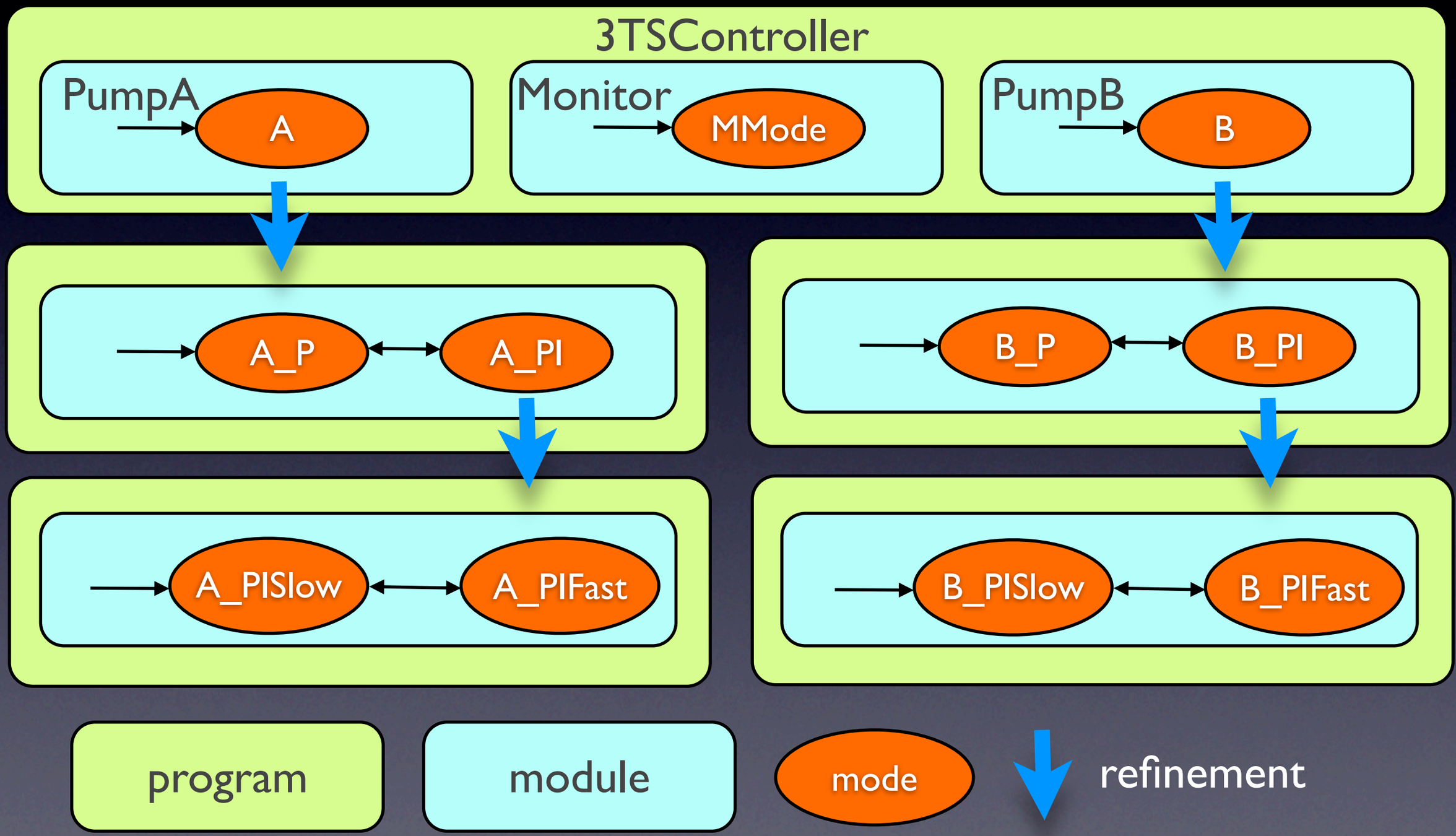
Hierarchical Timing Language

- HTL has four building blocks:
  - task (computation)
  - mode (precedences)
  - module (sequential composition)
  - program (concurrency, refinement)

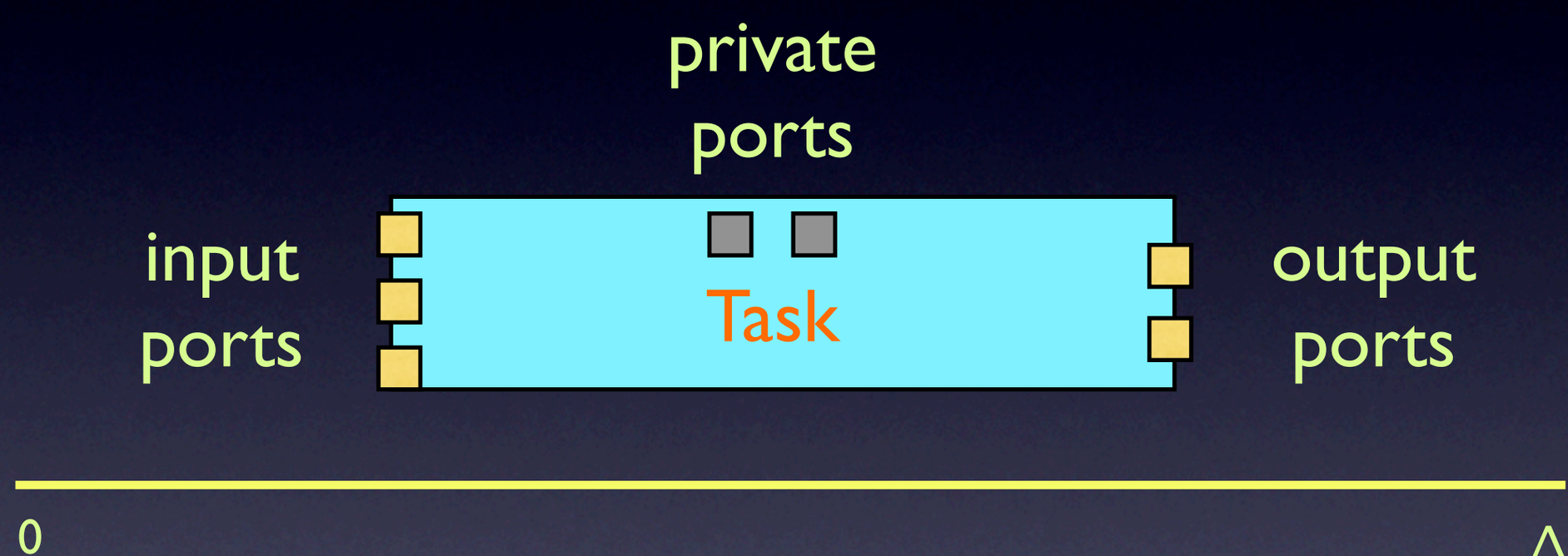


- an HTL program is an hierarchical, tree-like structure whose nodes are such blocks

Example

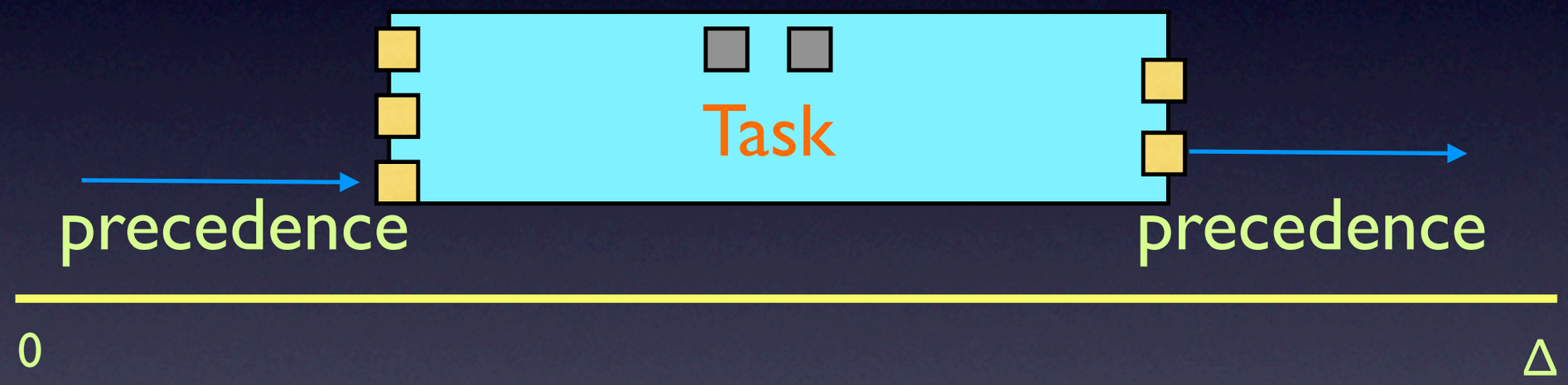

3TSController
PumpA
A
Monitor
MMode
PumpB
B
A_P
A_PI
B_P
B_PI
A_PISlow
A_PIFast
B_PISlow
B_PIFast
program
module
mode
refinement

Task



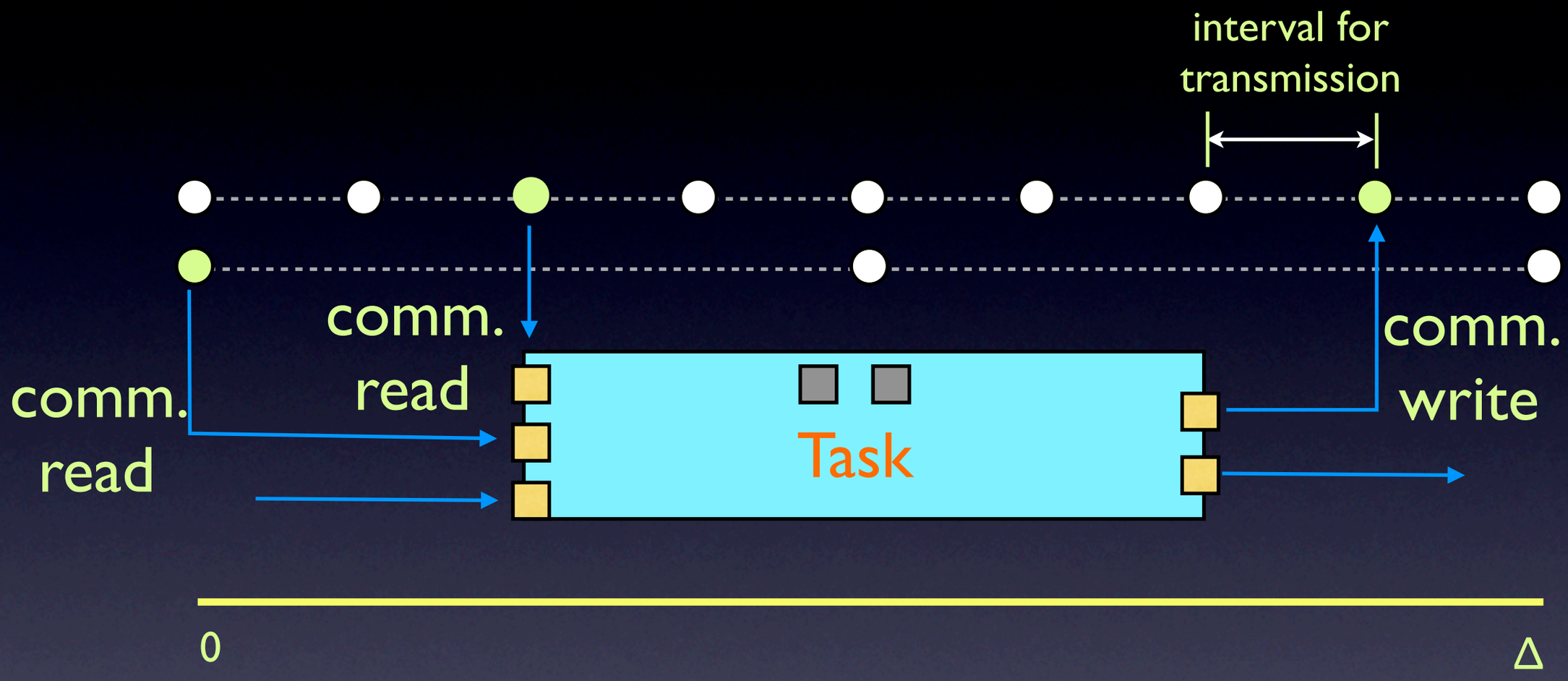
- an insulated code procedure (C / Java)
  - no side effects, no synchronization
  - executes periodically

Precedences



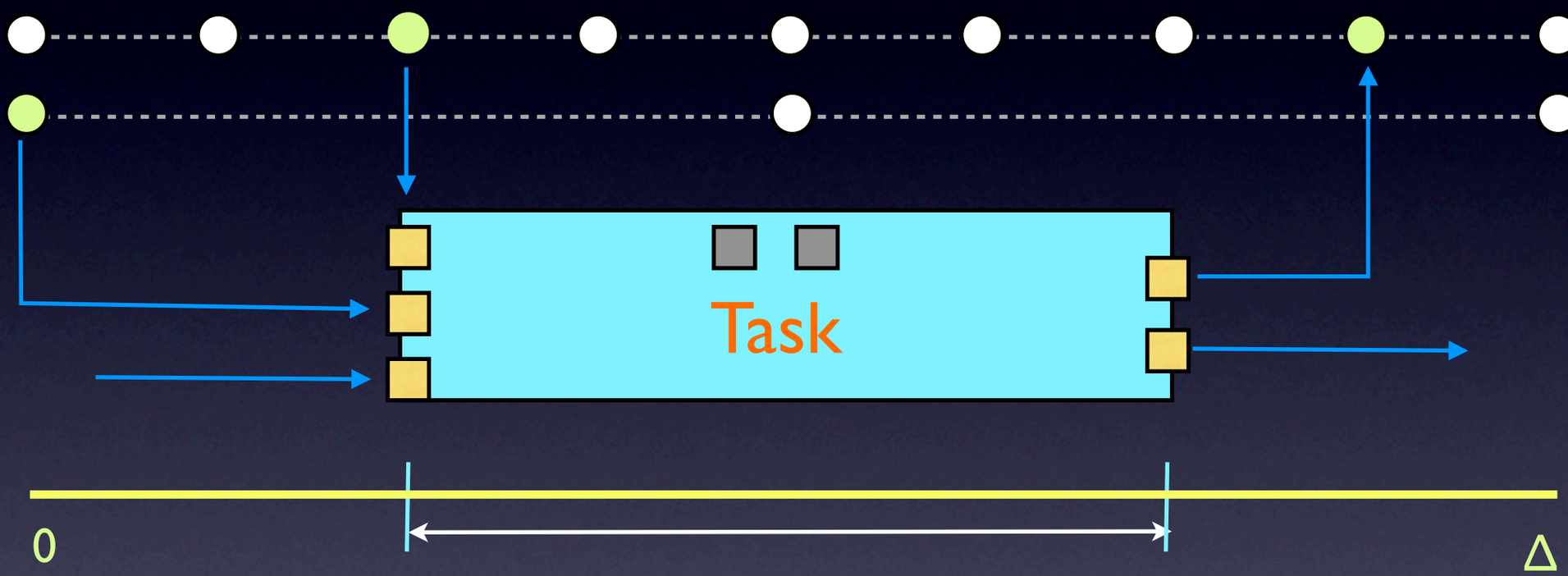
- tasks with equal periods may communicate through ports, defining task precedences

Communicators



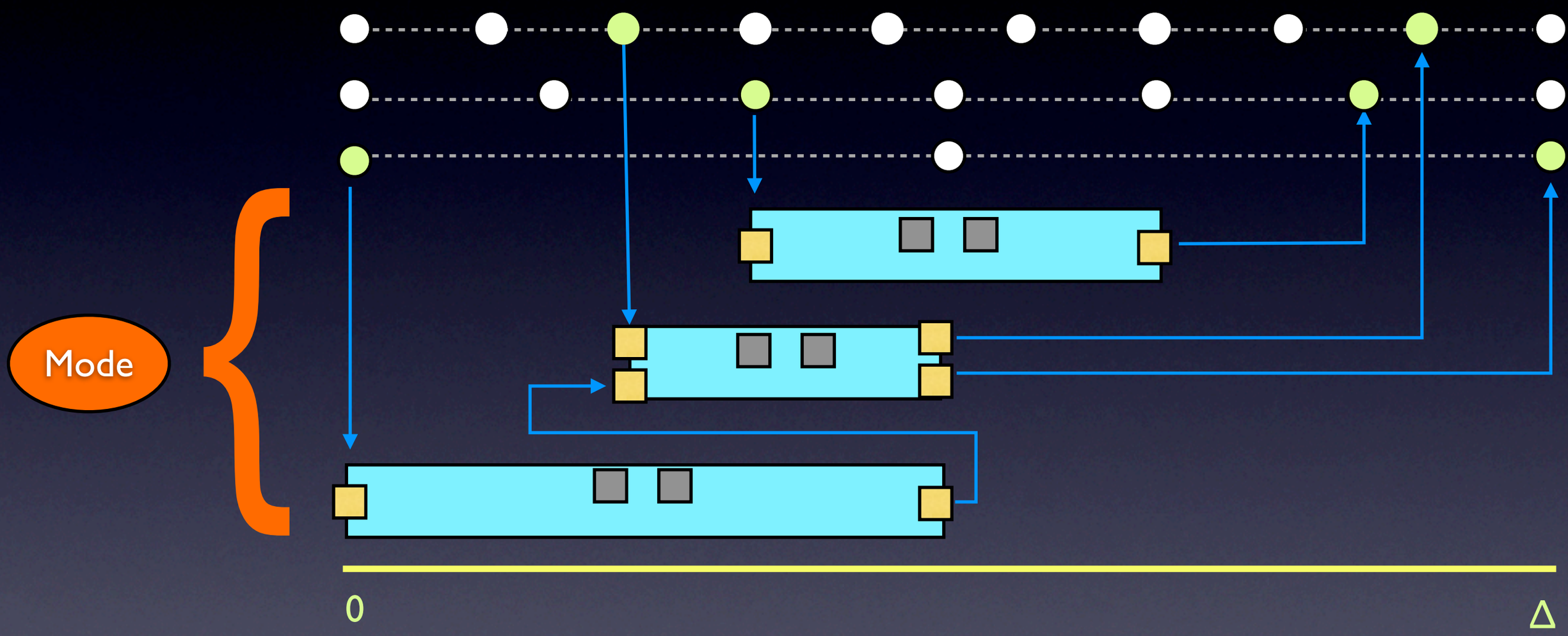
- a communicator is a periodically updated, program-wide variable
- tasks with different periods must communicate through communicators

Logical Execution Time



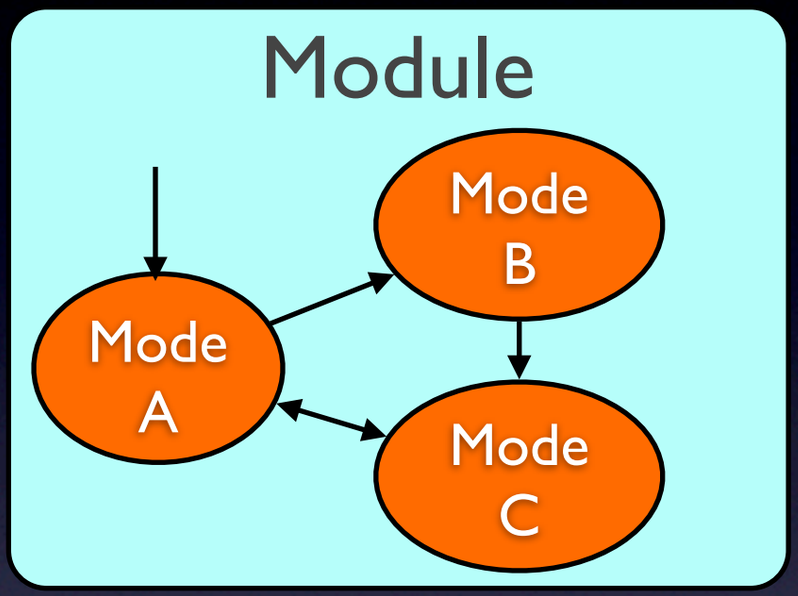
LET defined by communicator accesses and precedences

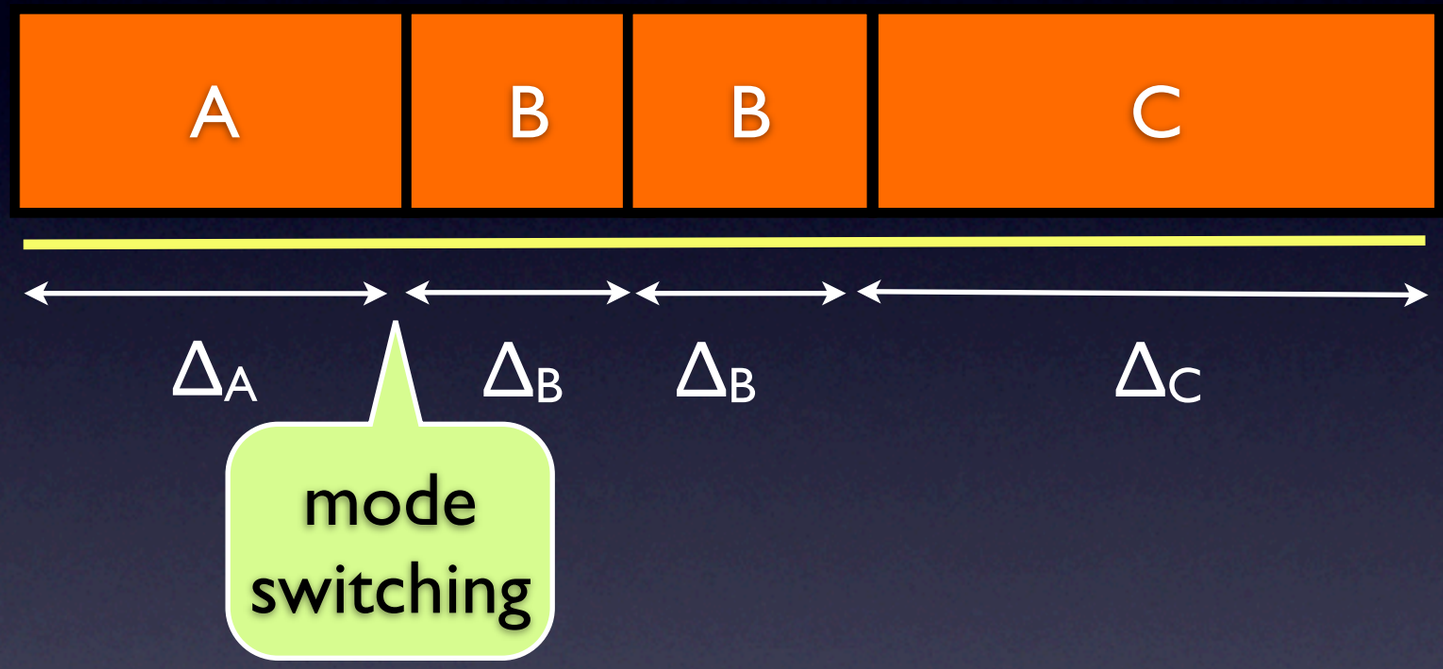
Mode



- a mode is a set of tasks with equal period and precedences between them.

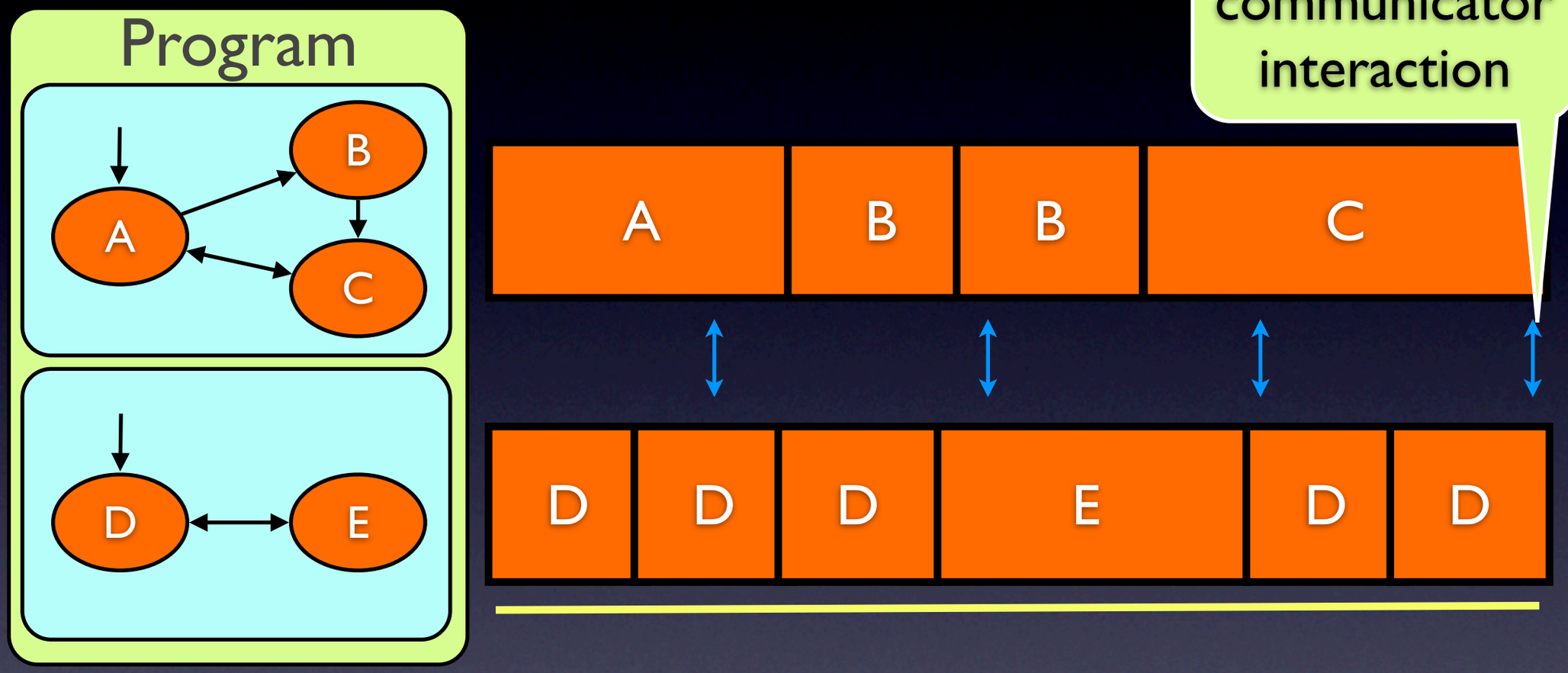
Module





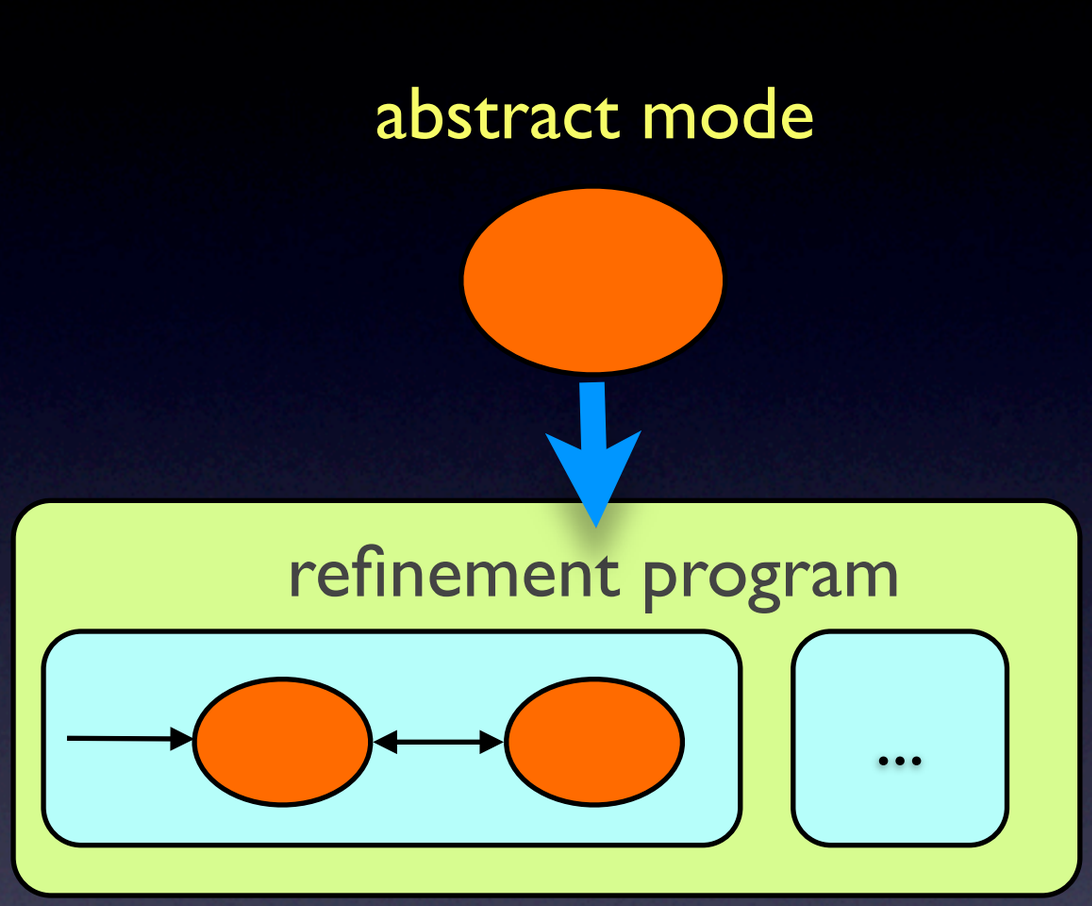
- a module is a set of modes that alternate execution according to a mode switching specification

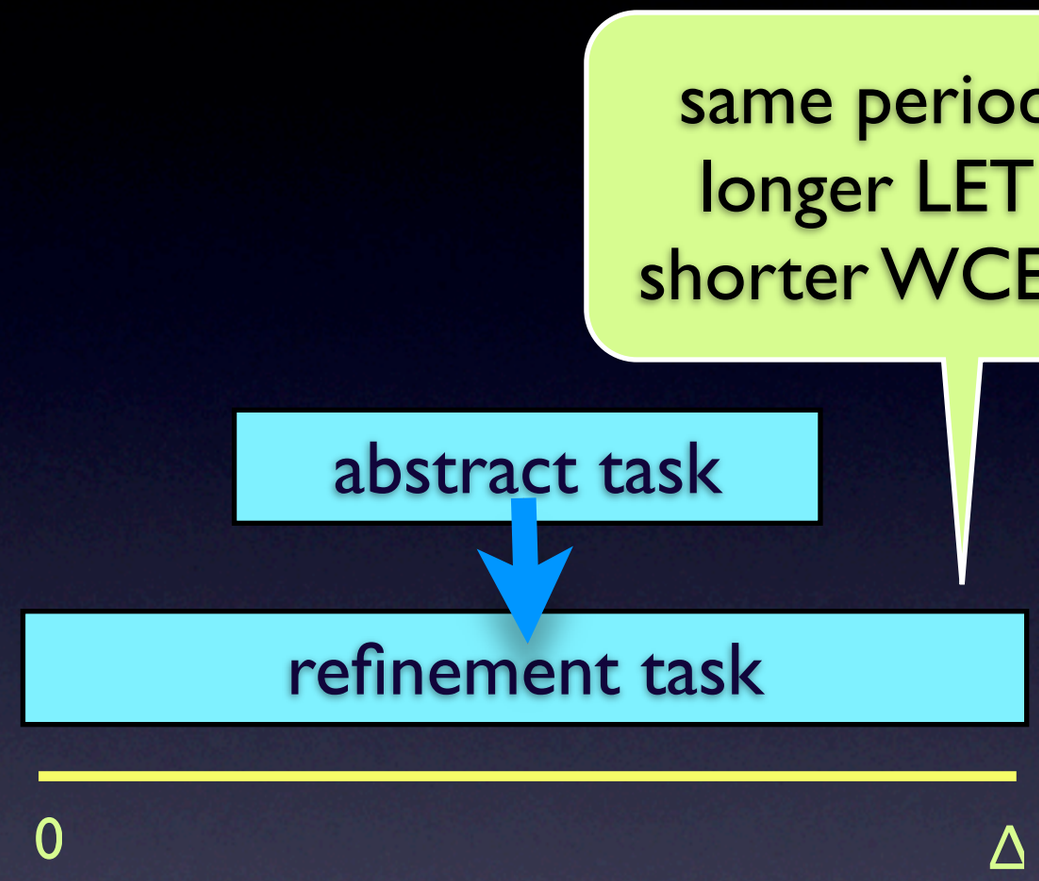
Program



- a program is a set of concurrent modules
- programs are distributed module-wise

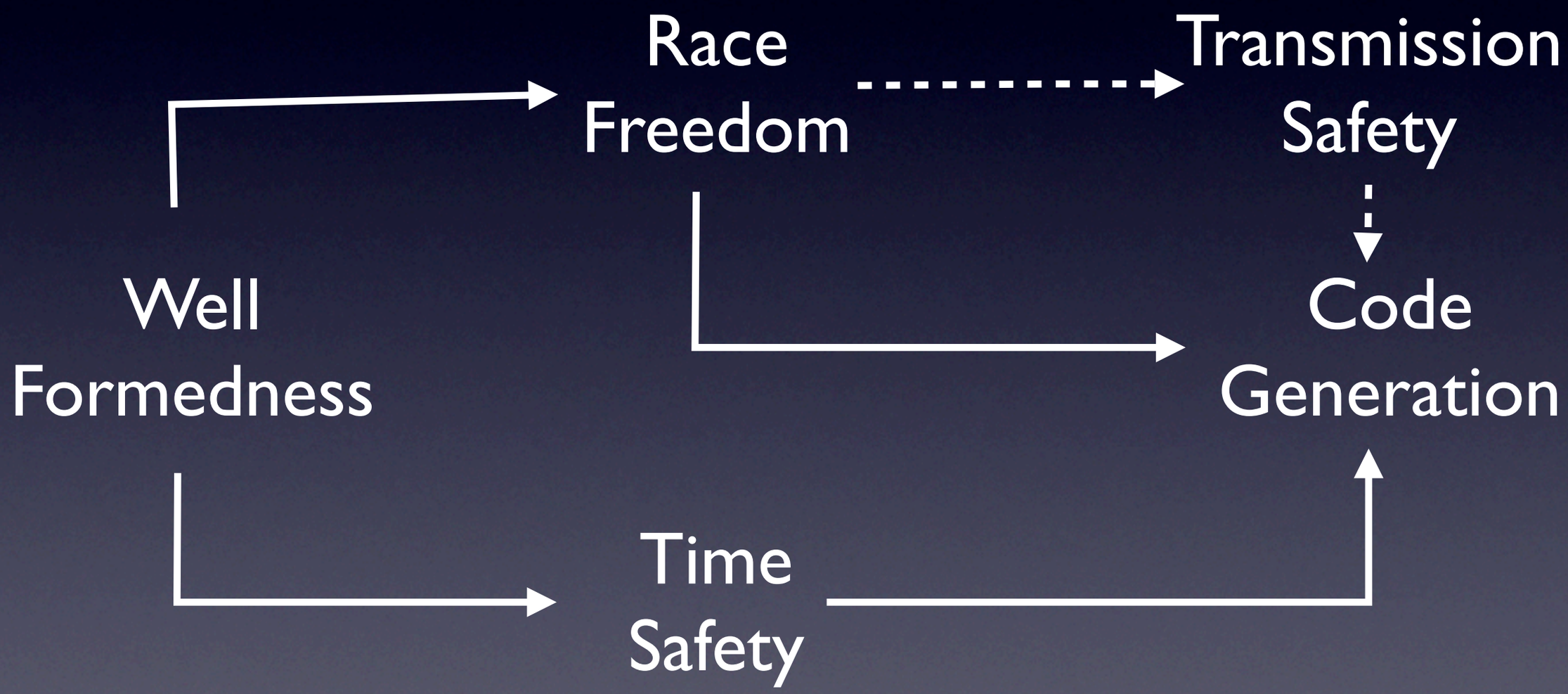
Refinement





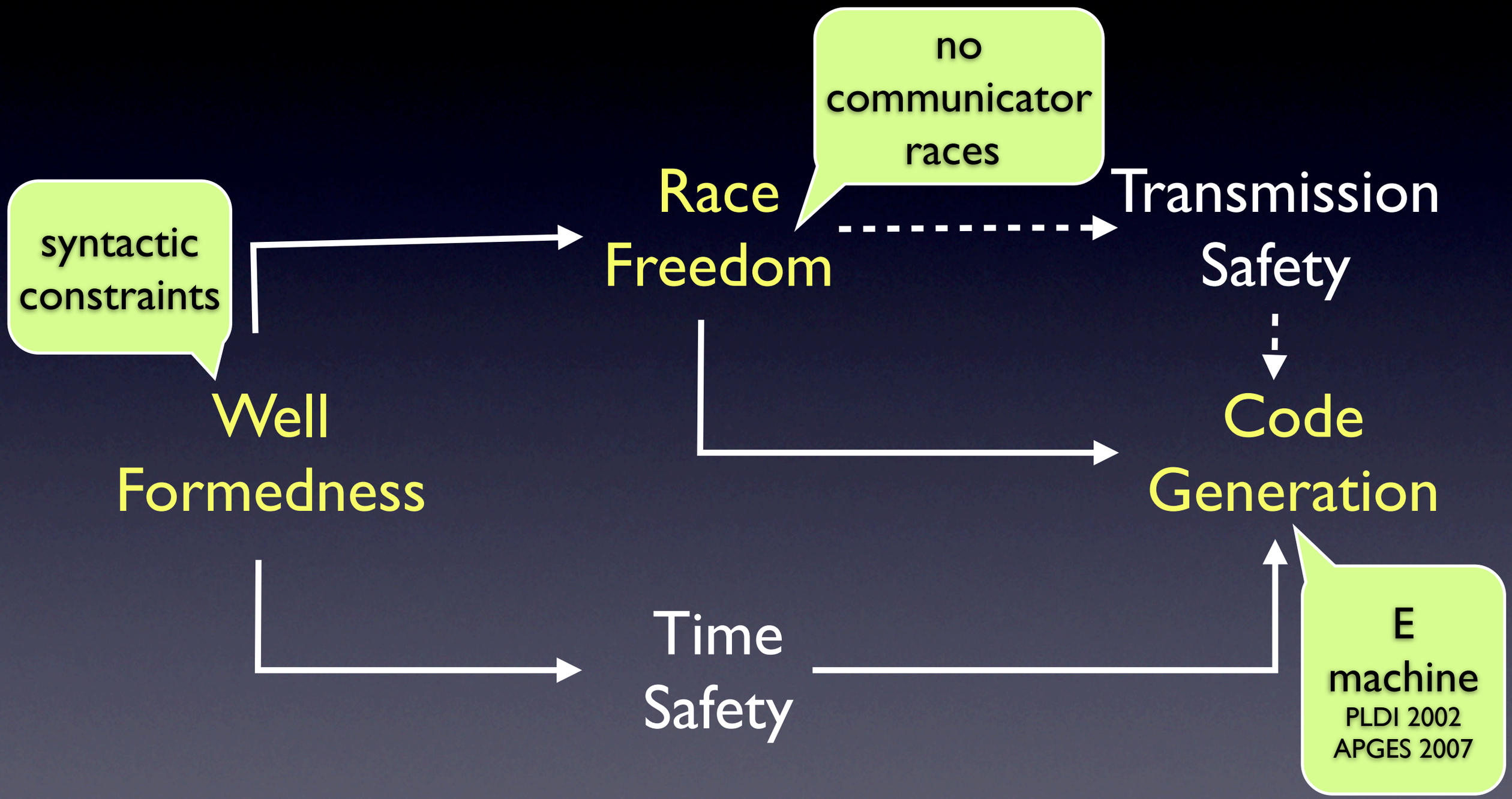
- modes can be refined
- refinement preserves abstract behavior

HTL Compilation

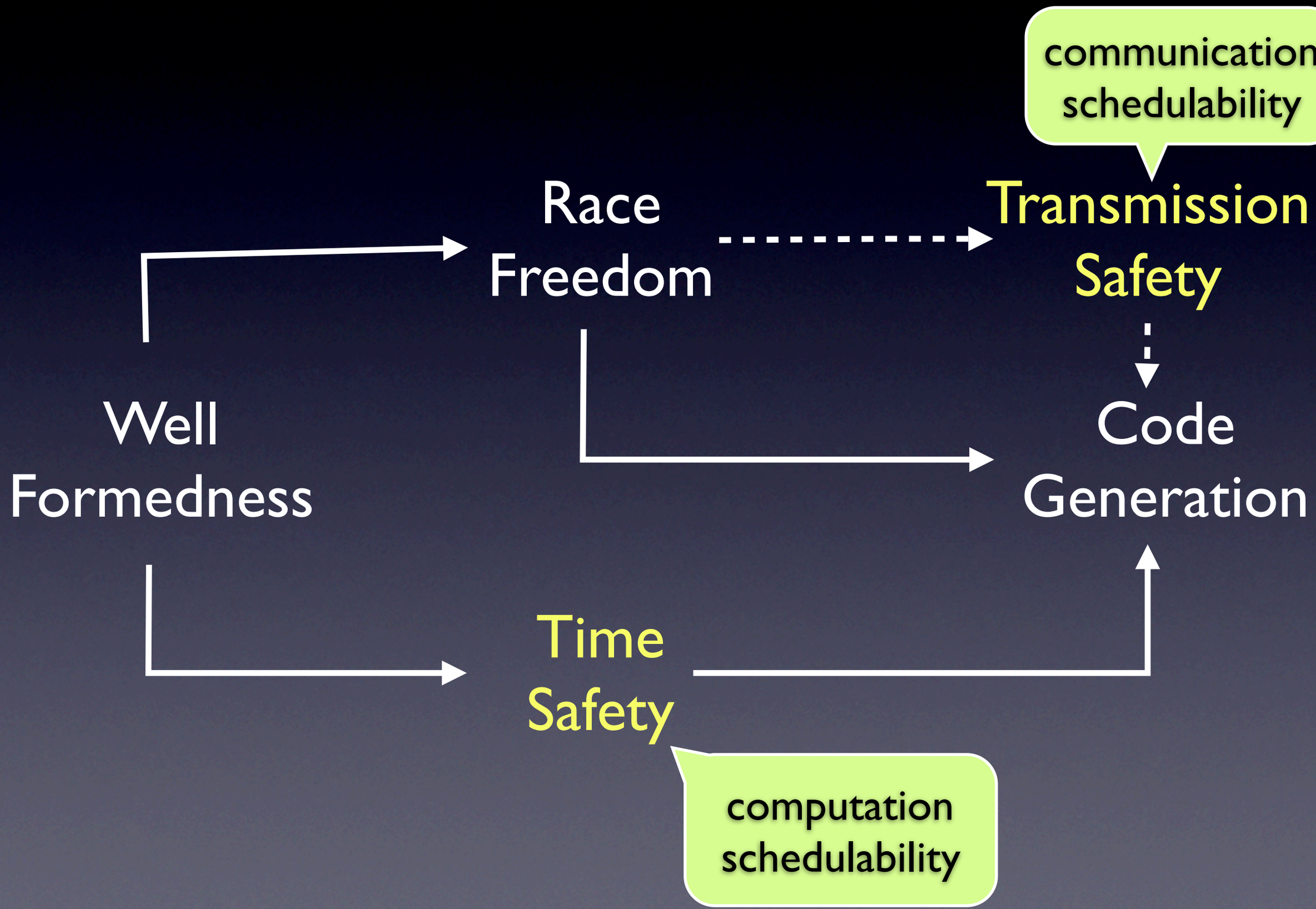


Well-formed, race-free, time-safe, and transmission-safe HTL programs are time-deterministic

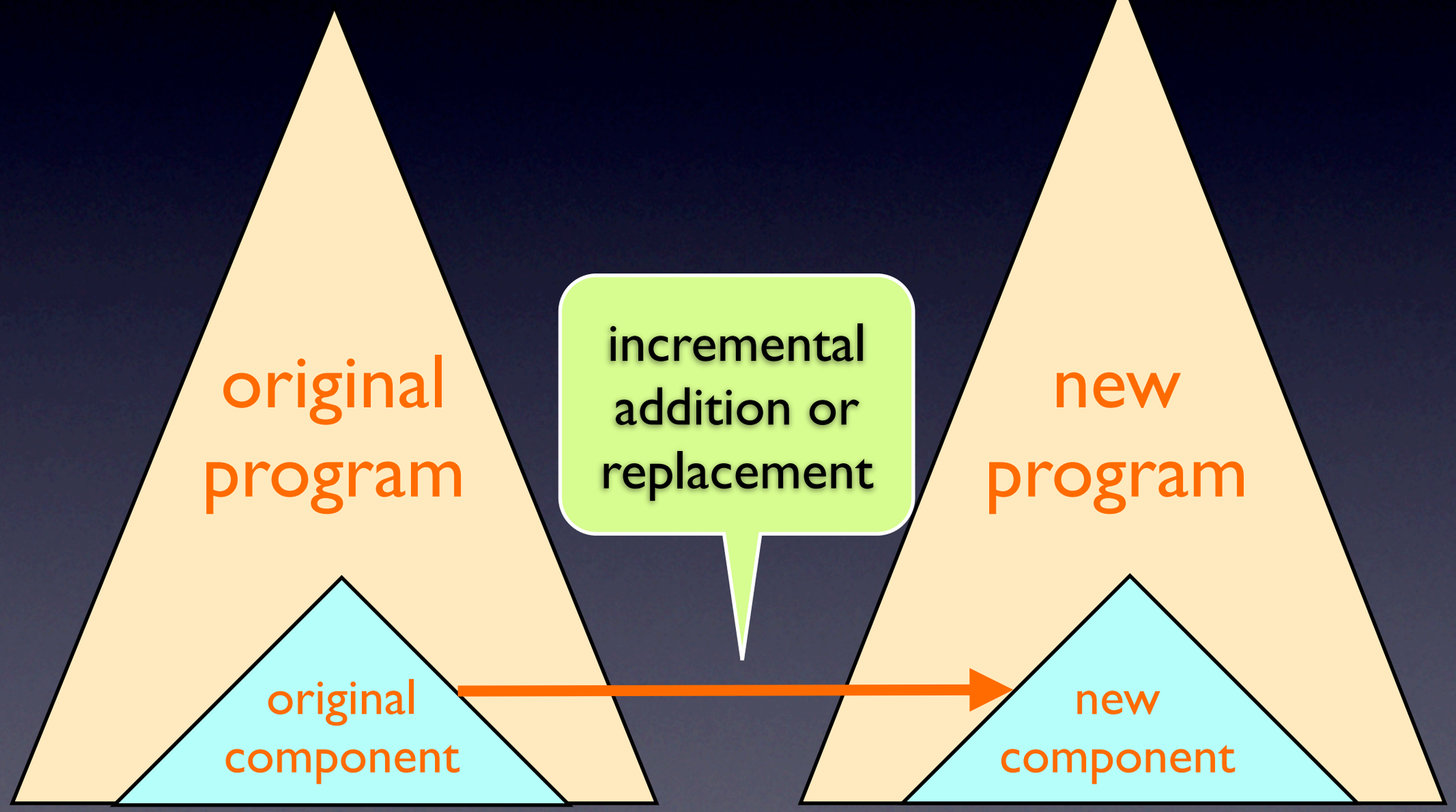
Platform-independent



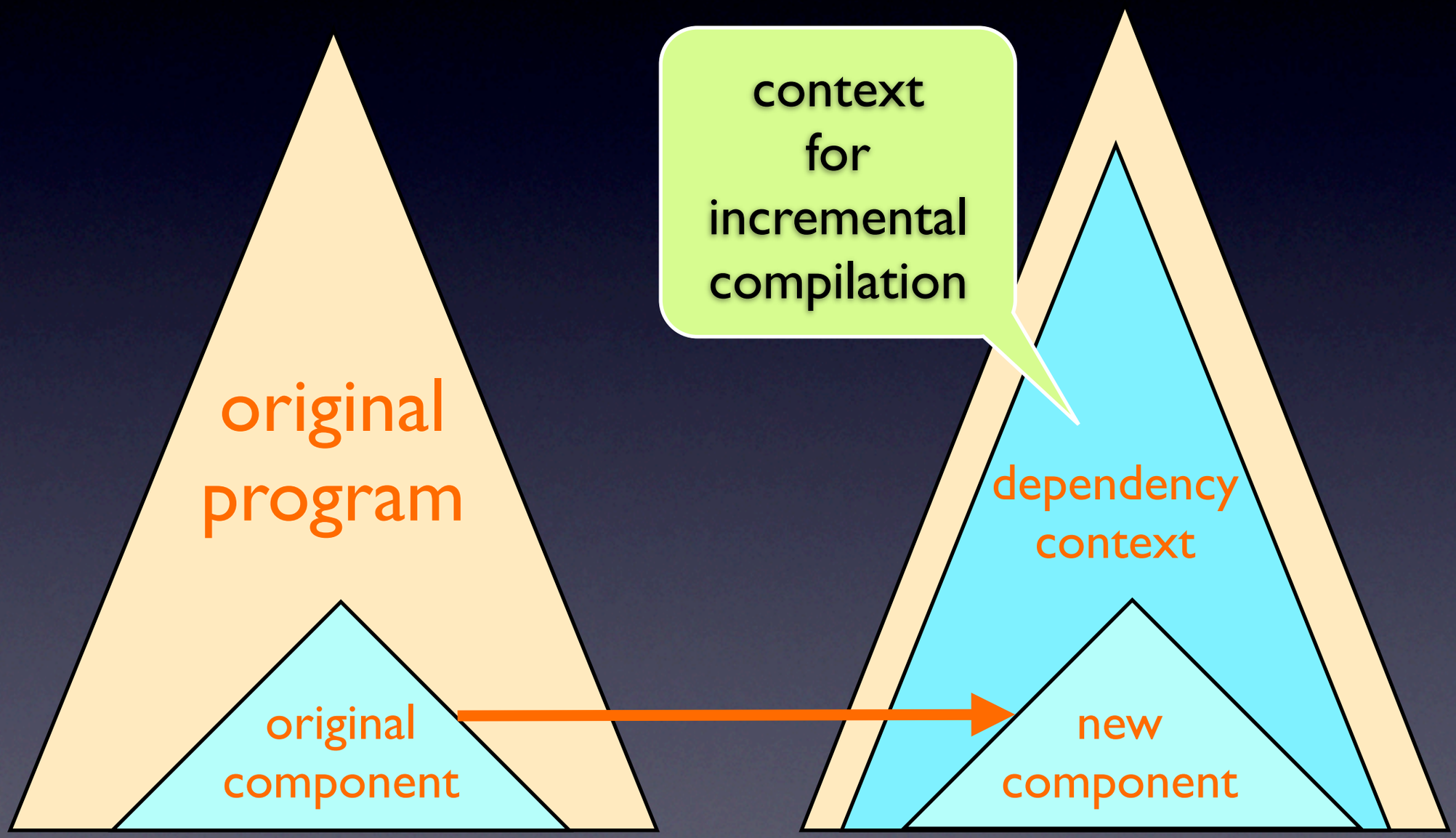
Platform-dependent



Modularity


original program
incremental addition or replacement
new program
original component
new component

Modularity


context
for
incremental
compilation
original
program
original
component
dependency
context
new
component

Modularity

Aspect	Component (C)	Dependency Context	Complexity
Well Formedness	any	C	linear
Race Freedom	top	P	linear
	refinement	C	no check
Time Safety	top	P	exponential
	refinement	C	no check
Transmission Safety	any	communicators	linear
Code generation	any	C	linear

P : full program

Modularity

Aspect	Component (C)	Dependency Context	Complexity
Well Formedness	any	C	linear
Race Freedom	top	P	linear
	refinement	C	no check
Time Safety	top	P	exponential
	refinement	C	no check
Transmiss Safety			
Code gene			

A concrete HTL program that refines a time-safe, race-free,abstract HTL program is also time-safe and race-free.

P : full program

Modularity

Aspect	Component (C)	Dependency Context	Complexity
Well Formedness	ar		linear
Race Freedom	to		linear
	refine		no check
Time Safety	top	P	exponential
	refinement	C	no check
Transmission Safety	any	communicators	linear
Code generation	any	C	linear

Fully non-modular.

P : full program

Modularity

Aspect	Component (C)	Dependency Context	Complexity
Well Formedness	any	C	linear
Race F	top	P	linear
			heck
Time			ential
			heck
Transmission safety	any	communicators	linear
Code generation	any	C	linear

Can be inferred solely from communicator periods and platform worst-case transmission times.

P : full program

Modularity

Aspect	Component (C)	Dependency Context	Complexity
Well Formedness	any	C	linear
Race Freedom	top	P	linear
	refinement	C	no check
Time Safety	top	P	exponential
Tran s			
Code generation	any	C	linear

Fully modular: code can be generated separately per component [APGES 2007].

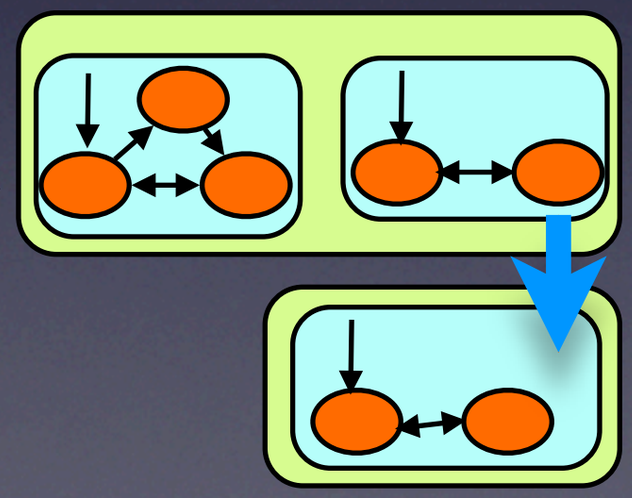
P : full program

Distribution

- transmission-safety can be asserted by standard schedulability criteria for a variety of network platforms (e.g. TDMA, FTT-CAN).
- time-safety analysis and code generation can be done separately per host
- overall: scalable distribution

Conclusion

- HTL = Hierarchical Timing Language
- Modularity = compositionality
- HTL is modular (syntax and semantics)
- HTL compilation is (quite) modular
- HTL distribution is modular



Conclusion

- HTL = Hierarchical Timing Language
- Mo
- HT
- HTL compilation is (quite) modular
- HTL distribution is modular

Thank you