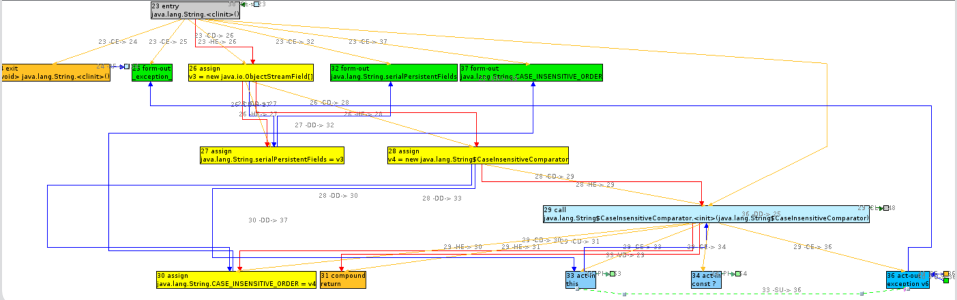
Slicing Concurrent Programs

Achievements and Open Challenges

Dennis Giffhorn
Programming paradigms group – IPD Snelting



Interference dependence

- Concurrency via threads and shared memory
- Shared-memory communication gives rise to *interference dependence*

Definition

Statement t is interference-dependent on statement s, if

- t uses a value which is defined by statement s, and
- s and t *may happen in parallel*

Interference dependence

- Concurrency via threads and shared memory
- Shared-memory communication gives rise to *interference dependence*

Definition

Statement t is interference-dependent on statement s, if

- t uses a value which is defined by statement s, and
- s and t *may happen in parallel*

Example

```
thread_1:
  a = input + c;
  b = input - 5;
```

```
thread_2:
  c = b;
```

Interference dependence

- Concurrency via threads and shared memory
- Shared-memory communication gives rise to *interference dependence*

Definition

Statement t is interference-dependent on statement s, if

- t uses a value which is defined by statement s, and
- s and t *may happen in parallel*

Example

```
thread_1:
  a = input + c;
  b = input - 5;
```

```
thread_2:
  c = b;
```

Interference dependence

- Concurrency via threads and shared memory
- Shared-memory communication gives rise to *interference dependence*

Definition

Statement t is interference-dependent on statement s, if

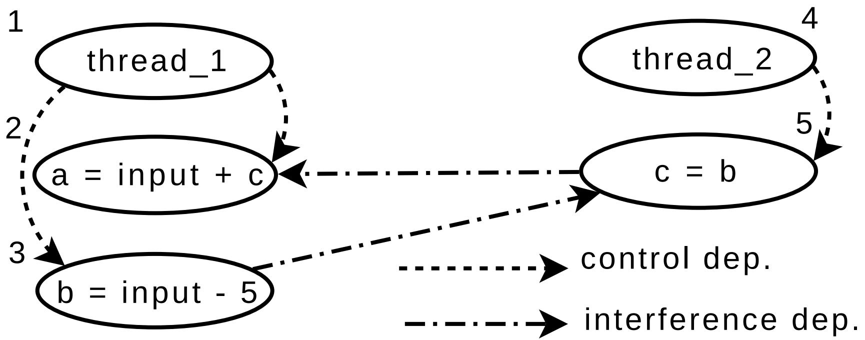
- t uses a value which is defined by statement s, and
- s and t *may happen in parallel*

Example

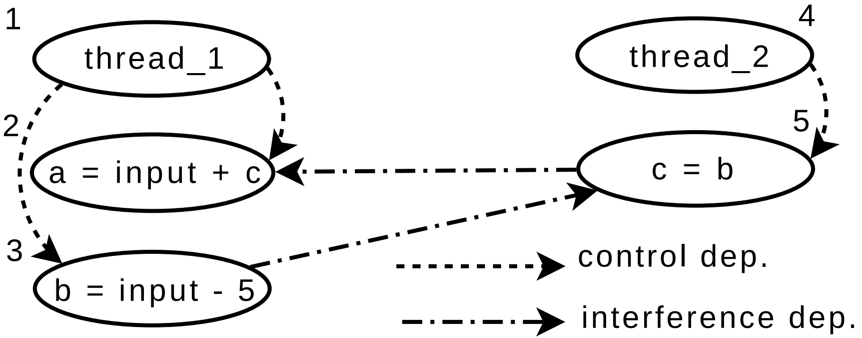
```
thread_1:
  a = input + c;
  b = input - 5;
```

```
thread_2:
  c = b;
```

cSDG – concurrent SDG

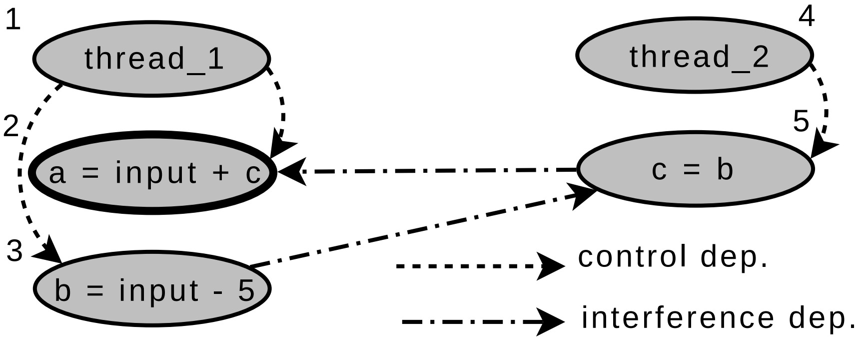


cSDG – concurrent SDG



- Two interference dependences may exclude each other

cSDG – concurrent SDG



- Two interference dependences may exclude each other

⇒ Time-insensitive slices

Time-sensitive Slicing

Karlsruhe Institute of Technology

Time-sensitive path

A context-sensitive path $\langle n_1, \ldots, n_k \rangle$ in a cSDG is time-sensitive, if $\forall 1 \leq i < j \leq k$:

- n_i and n_j may happen in parallel, or
- n_i reaches n_j in the control flow graph

Time-sensitive Slicing

Time-sensitive path

A context-sensitive path $\langle n_1, \ldots, n_k \rangle$ in a cSDG is time-sensitive, if $\forall 1 \leq i < j \leq k$:

- n_i and n_j may happen in parallel, or
- n_i reaches n_j in the control flow graph

Prepending property (Krinke, 2003)

Let $\pi = \langle n_1, \ldots, n_k \rangle$ be a time-sensitive path in a cSDG G. Let $e = n_0 \rightarrow n_1$ be an edge in G. Path $\langle n_0, n_1, \ldots, n_k \rangle$ is time-sensitive, if

- e is thread-local, or
- n_0 reaches the first element n in π that may not happen in parallel to n_0

Time-sensitive Slicing

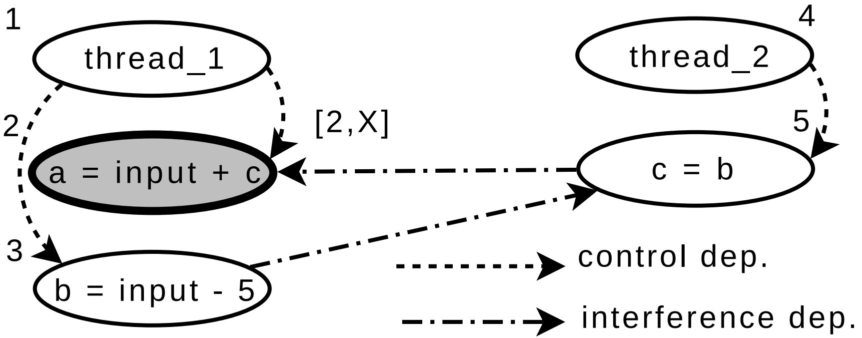
Exploit prepending property:

- Annotate nodes with *state tuples*

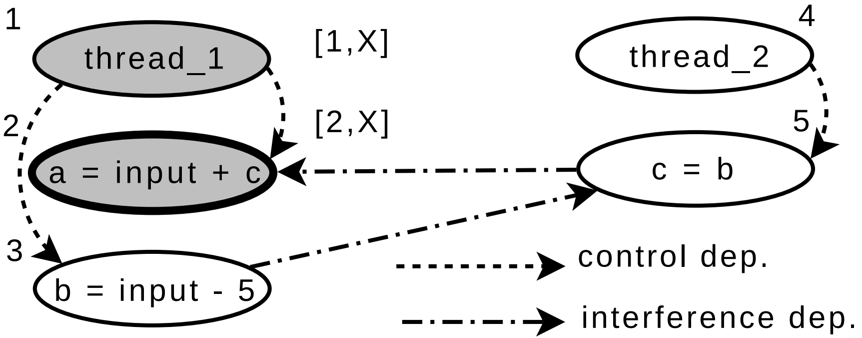
→ One entry per thread

→ Contains the first element of that thread in the path taken so far

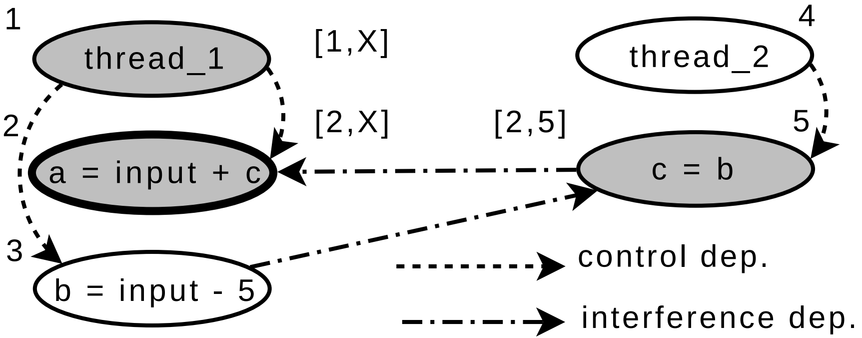
Time-sensitive Slicing



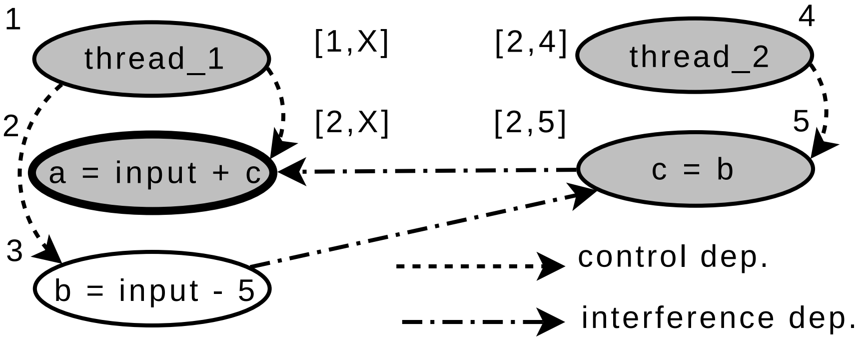
Time-sensitive Slicing



Time-sensitive Slicing



Time-sensitive Slicing



Time-sensitive Slicing

Runtime costs

- A node can be visited repeatedly with different state tuples
- State tuple entry is node + calling context

$\Rightarrow$ $\mathcal{O}(N^{p^t})$,
N = number of nodes,
p = max. call depth,
N^p = upper bound for nodes + calling contexts,
t = number of threads

Time-sensitive Slicing

Karlsruhe Institute of Technology

- Practical for programs with approx. 10 kLoc
- Usable for mature languages
  - Interprocedural programs, including recursion
  - Dynamic thread creation inside loops or recursion
- JOANA-Project for full Java bytecode
  (Giffhorn and Hammer, *Precise Analysis of Java Programs using JOANA (Tool Demonstration)*, in *8th IEEE SCAM*, 2008.

Empirical Results

name	nodes	edges	methods	threads	interf. dep.
DiningPhils	5143	125470	116	2	471
LaplaceGrid	6218	51035	161	2	948
Barcode	12393	64820	271	2	5
HyperM	17835	97827	277	6	8139
Podcast	23676	159478	407	3	128

- 1,000 slices per program:

name	size per slice (nodes)		runtime per slice (msec)	
	cont.-sens.	time-sens.	cont.-sens.	time-sens.
DiningPhils	2,867	2,499 (87%)	25	6,711
LaplaceGrid	3,409	3,328 (98%)	12	10,167
Barcode	3,410	2,974 (87%)	12	275
HyperM	9,222	7,441 (81%)	36	2,888
Podcast	12,335	8,730 (71%)	50	7,286

Important Topics for Future Research

- Runtime costs
- Prepending property and MHP information
- Interference dependence and reaching definitions

Runtime Costs

- What property of a cSDG has the major influence on the runtime costs?

name	nodes	edges	methods	threads	interf. dep.
DiningPhils	5143	125470	116	2	471
LaplaceGrid	6218	51035	161	2	948
Barcode	12393	64820	271	2	5
HyperM	17835	97827	277	6	8139
Podcast	23676	159478	407	3	128

name	size per slice (nodes)		runtime per slice (msec)	
	cont.-sens.	time-sens.	cont.-sens.	time-sens.
DiningPhils	2,867	2,499 (87%)	25	6,711
LaplaceGrid	3,409	3,328 (98%)	12	10,167
Barcode	3,410	2,974 (87%)	12	275
HyperM	9,222	7,441 (81%)	36	2,888
Podcast	12,335	8,730 (71%)	50	7,286

Runtime Costs

Karlsruhe Institute of Technology

- What property of a cSDG has the major influence on the runtime costs?

name	nodes	edges	methods	threads	interf. dep.
DiningPhils	5143	125470	116	2	471
LaplaceGrid	6218	51035	161	2	948
Barcode	12393	64820	271	2	5
HyperM	17835	97827	277	6	8139
Podcast	23676	159478	407	3	128

name	size per slice (nodes)		runtime per slice (msec)	
	cont.-sens.	time-sens.	cont.-sens.	time-sens.
DiningPhils	2,867	2,499 (87%)	25	6,711
LaplaceGrid	3,409	3,328 (98%)	12	10,167
Barcode	3,410	2,974 (87%)	12	275
HyperM	9,222	7,441 (81%)	36	2,888
Podcast	12,335	8,730 (71%)	50	7,286

Runtime Costs

Karlsruhe Institute of Technology

- What property of a cSDG has the major influence on the runtime costs?

name	nodes	edges	methods	threads	interf. dep.
DiningPhils	5143	125470	116	2	471
LaplaceGrid	6218	51035	161	2	948
Barcode	12393	64820	271	2	5
HyperM	17835	97827	277	6	8139
Podcast	23676	159478	407	3	128

name	size per slice (nodes)		runtime per slice (msec)	
	cont.-sens.	time-sens.	cont.-sens.	time-sens.
DiningPhils	2,867	2,499 (87%)	25	6,711
LaplaceGrid	3,409	3,328 (98%)	12	10,167
Barcode	3,410	2,974 (87%)	12	275
HyperM	9,222	7,441 (81%)	36	2,888
Podcast	12,335	8,730 (71%)	50	7,286

Runtime Costs

- What property of a cSDG has the major influence on the runtime costs?

name	nodes	edges	methods	threads	interf. dep.
DiningPhils	5143	125470	116	2	471
LaplaceGrid	6218	51035	161	2	948
Barcode	12393	64820	271	2	5
HyperM	17835	97827	277	6	8139
Podcast	23676	159478	407	3	128

name	size per slice (nodes)		runtime per slice (msec)	
	cont.-sens.	time-sens.	cont.-sens.	time-sens.
DiningPhils	2,867	2,499 (87%)	25	6,711
LaplaceGrid	3,409	3,328 (98%)	12	10,167
Barcode	3,410	2,974 (87%)	12	275
HyperM	9,222	7,441 (81%)	36	2,888
Podcast	12,335	8,730 (71%)	50	7,286

Prepending Property and MHP Information

- May-happen-in-parallel analysis:
  Which parts of the threads may happen in parallel?
- Recall the definition of time-sensitive paths:

Time-sensitive path

A context-sensitive path $\langle n_1, \ldots, n_k \rangle$ in a cSDG is time-sensitive, if $\forall 1 \leq i < j \leq k$:

- n_i and n_j may happen in parallel, or
- n_i reaches n_j in the control flow graph

- Time-sensitivity depends on available MHP information
- In our evaluation it increased precision by 10%

Prepending Property and MHP Information

- May-happen-in-parallel analysis:
  Which parts of the threads may happen in parallel?
- Recall the definition of time-sensitive paths:

Time-sensitive path

A context-sensitive path $\langle n_1, \ldots, n_k \rangle$ in a cSDG is time-sensitive, if $\forall 1 \leq i < j \leq k$:

- n_i and n_j may happen in parallel, or
- n_i reaches n_j in the control flow graph

- Time-sensitivity depends on available MHP information
- In our evaluation it increased precision by 10%
- Cannot be completely exploited by the time-sensitive slicer

Prepending Property and MHP Information

Karlsruhe Institute of Technology

Prepending property (Krinke, 2003)

Let $\pi = \langle n_1, \ldots, n_k \rangle$ be a time-sensitive path in a cSDG G. Let $e = n_0 \rightarrow n_1$ be an edge in G. Path $\langle n_0, n_1, \ldots, n_k \rangle$ is time-sensitive, if

- e is thread-local, or
- n_0 reaches the first element n in π that may not happen in parallel to n_0

- Does only hold in case all threads may happen in parallel
  - $\Rightarrow$ n_0 and n may not happen in parallel to the same elements M in π
  - $\Rightarrow$ n reaches every $m \in M$
  - $\Rightarrow$ 'Reaches' is transitive, thus it suffices that n_0 reaches n

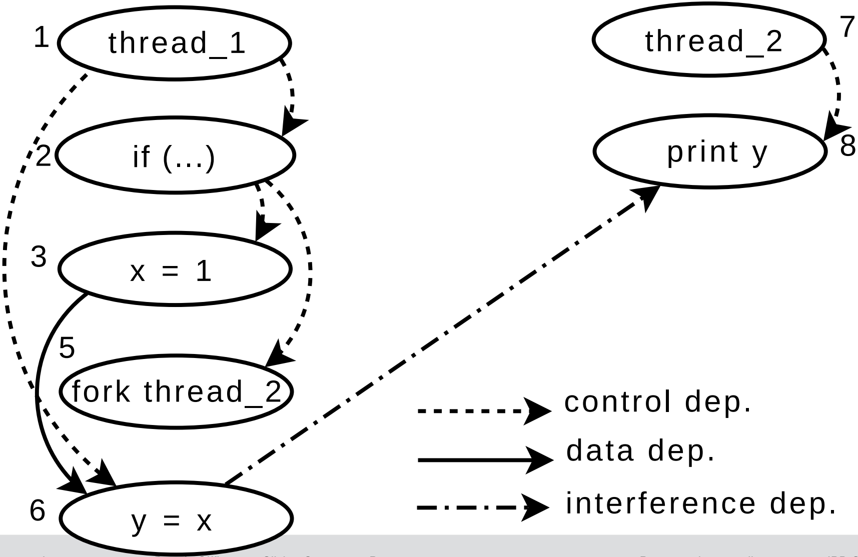
Prepending Property and MHP Information

Example

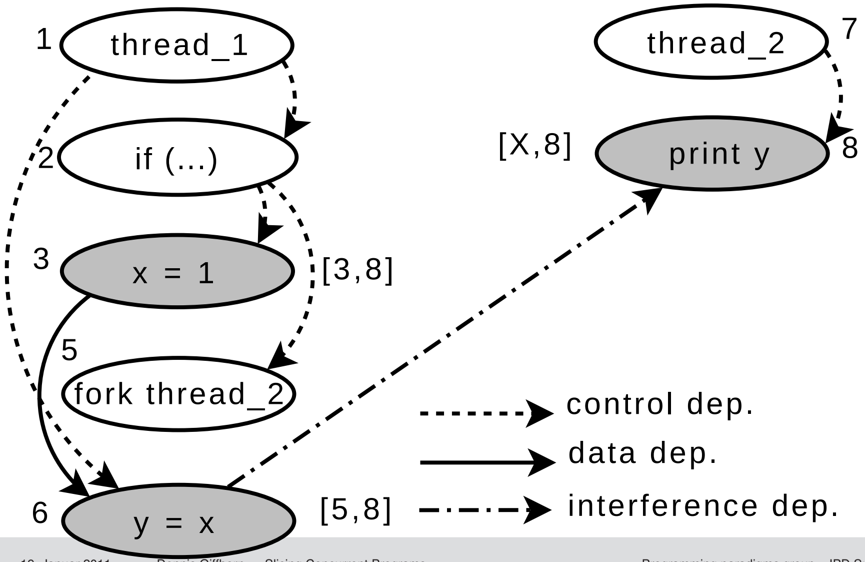
```
thread_1:
  if (...)
    x = 1;
  else
    fork thread_2;
  y = x;

thread_2:
  print y;
```

Prepending Property and MHP Information



Prepending Property and MHP Information



Prepending Property and MHP Information

Develop a slicer that is time-sensitive wrt. general MHP information

- Adjust the prepending property?
  - ⇒ Which information has to be stored in the state tuples?
- Find a completely different approach?
  - Post-process time-insensitive slices
  - Deactivate invalid parts of the cSDG (use Petri nets?)
  - . . .

Interference Dependence and Reaching Definitions

Karlsruhe Institute of Technology

Definition

Statement t is interference-dependent on statement s, if

- t uses a variable v which is defined by statement s, and
- s and t *may happen in parallel*

- Data dependence requires reaching definitions
- Interference dep. ignores reaching definitions
- Undecidable for concurrent interprocedural programs (Müller-Olm and Seidl, *On optimal slicing of parallel programs*, in *ACM Symposium on Theory of Computing*, 2001)
- Decidable in special cases

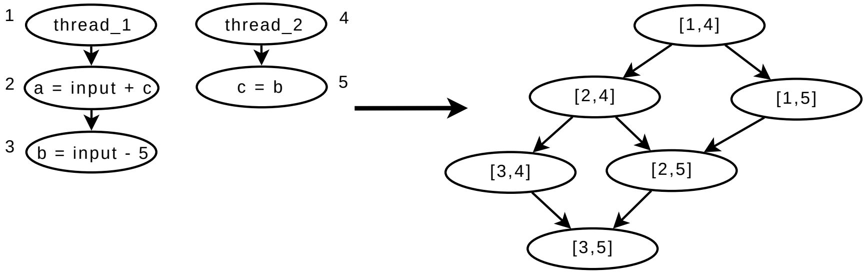
Threaded Interaction Reachability Graph

Karlsruhe Institute of Technology

Idea of Qi et al.
(*Slicing Concurrent Programs Based on Program Reachability Graphs*, in *IEEE 10th ICQS*, 2010.)

- Unroll the possible interleavings in a Threaded Interaction Reachability Graph (TIRG)
- Each node corresponds to a possible interleaving situation

Threaded Interaction Reachability Graph



Threaded Interaction Reachability Graph

- Create a SDG from the TIRG
  - ⇒ Data flow analysis for sequential programs
- Free of interference dependence
  - No time-insensitivity
  - Finds situations, where a def in one thread does not reach a use in another thread

Threaded Interaction Reachability Graph

- Create a SDG from the TIRG
  - ⇒ Data flow analysis for sequential programs
- Free of interference dependence
  - No time-insensitivity
  - Finds situations, where a def in one thread does not reach a use in another thread
- Very high precision
- No empirical data ⇒ assumption: TIRGs are huge (need to inline procedures)
- No recursion
- No thread creation inside loops

⇒ Extend its power

⇒ Develop compression techniques

Literature

- J. Krinke, *Context-sensitive Slicing of Concurrent Programs*, in *11th Foundations of Software Engineering*, 2003
- M. G. Nanda and S. Ramesh, *Interprocedural Slicing of Multithreaded Programs with Applications to Java*, in *ACM TOPLAS*, 28(6):1088–1144, 2006.
- D. Giffhorn and C. Hammer, *Precise Slicing of Concurrent Programs - An Evaluation of Static Slicing Algorithms for Concurrent Programs*, in *Springer JASE*, 16(2):197–234, 2009.

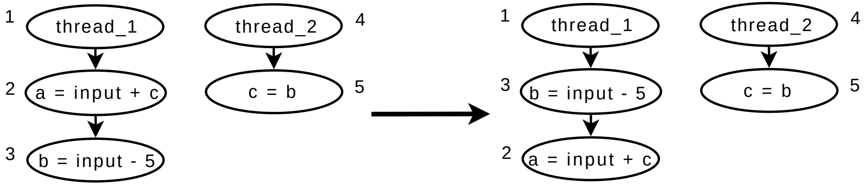
JIT Compiler

Karlsruhe Institute of Technology

- Just-in-time (JIT) compiler may reorganize code during program execution
- Control flow during execution may not correspond to the control flow during the slice

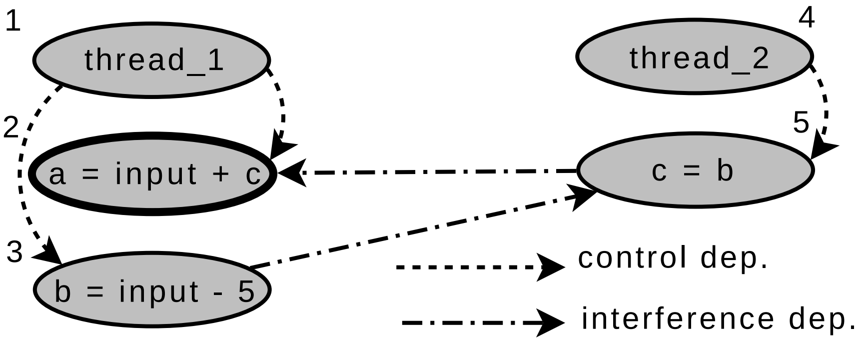
⇒ Time-insensitive paths may turn time-sensitive

JIT Compiler



- Java's JIT compiler may switch statements 2 and 3

JIT Compiler



- Slice is time-sensitive!
- Identification of which execution orders are guaranteed