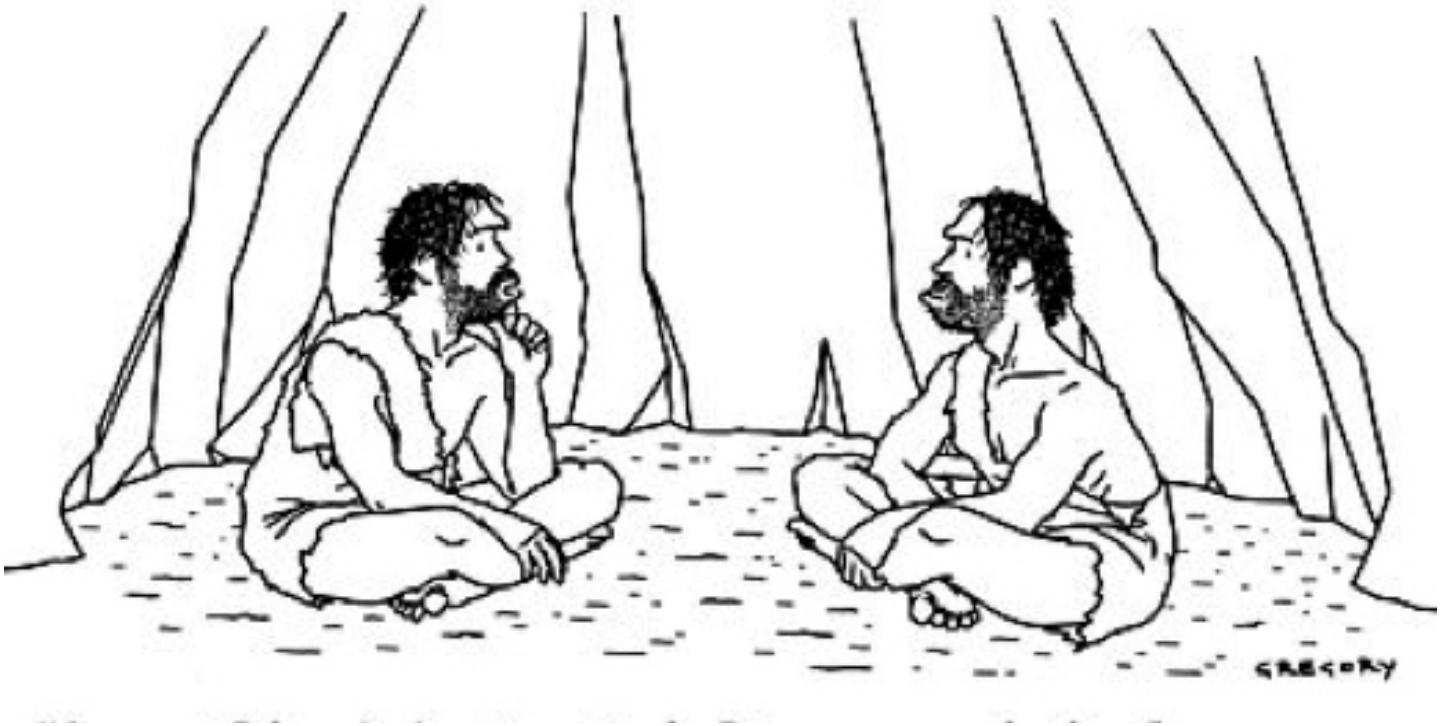
Why systems fail?
- blue screen, leaked pictures, dictators, cancer -
How to stop them?

Ashutosh Gupta

TIFR, India

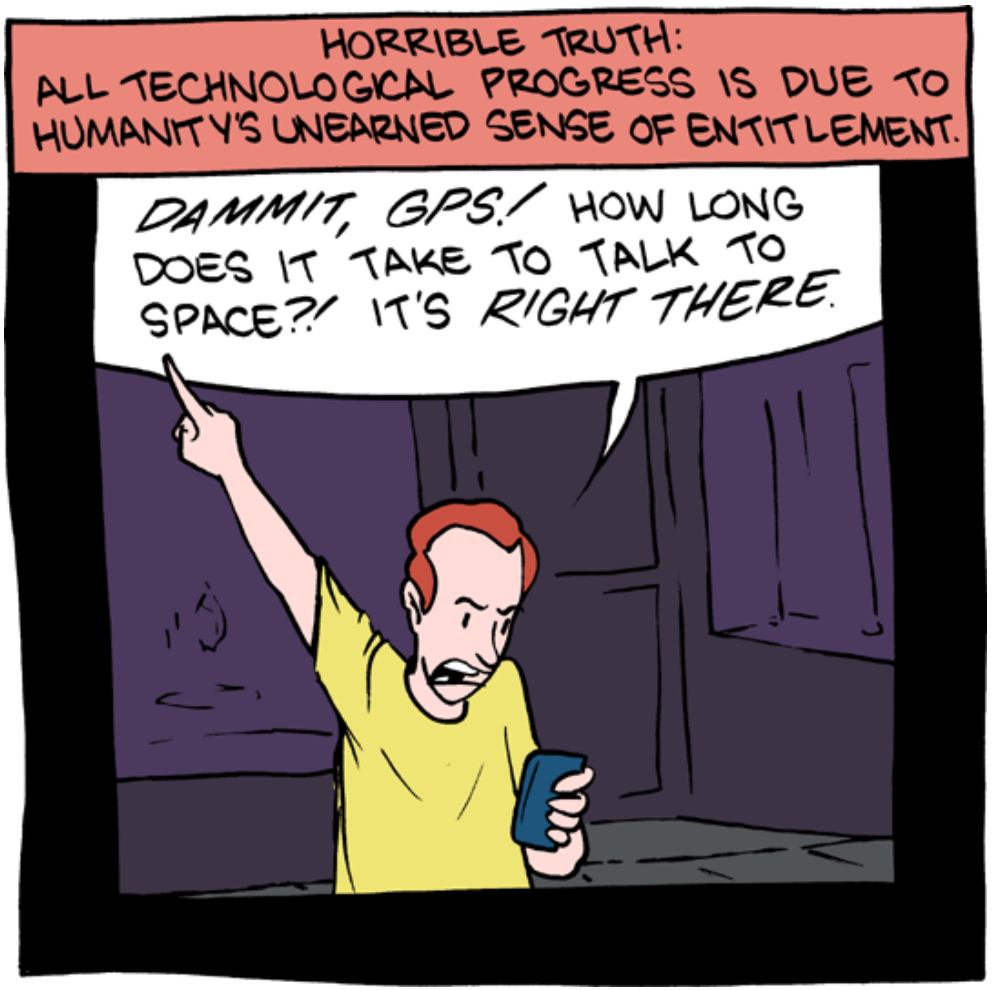
Compile date: 2016-03-18

There was a simple life



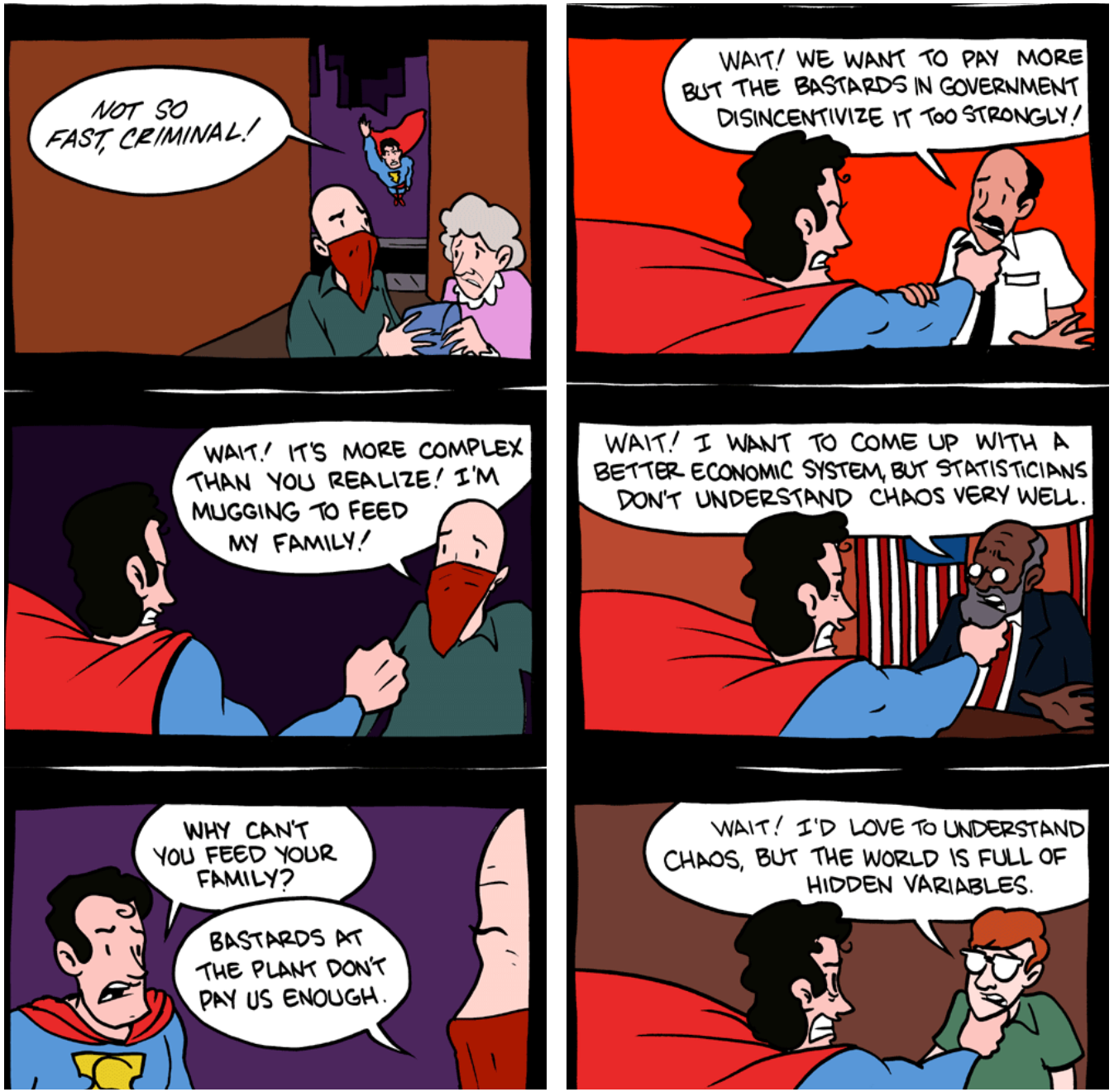
"Something's not right – our air and water is clean, we get plenty of exercise, everything we eat is organic and free-range, and yet nobody lives past thirty"

And progress happened

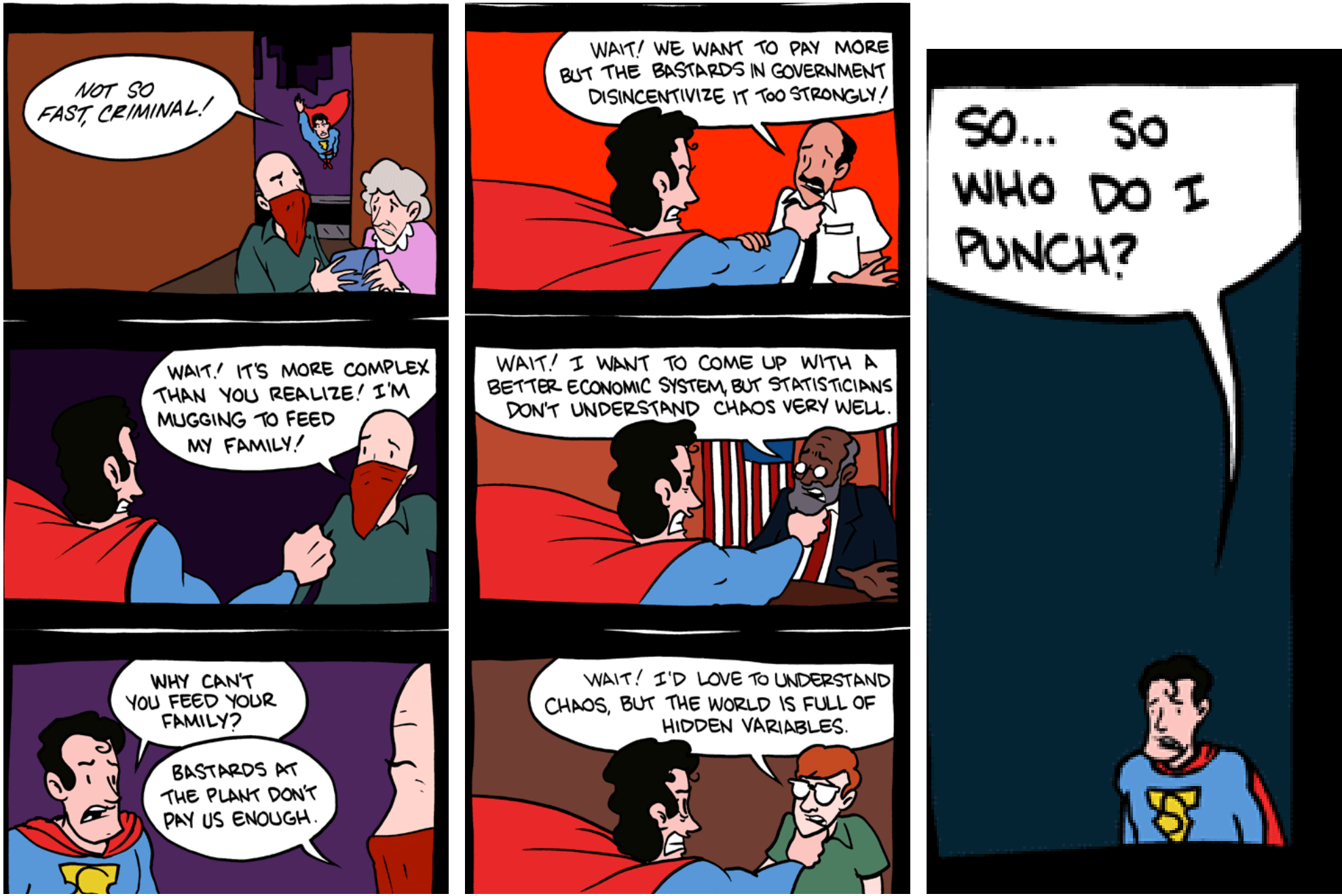


Progress brought complexity

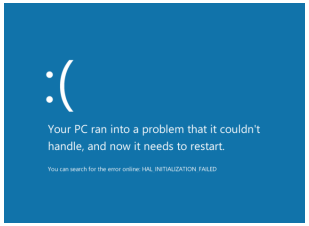
Progress brought complexity



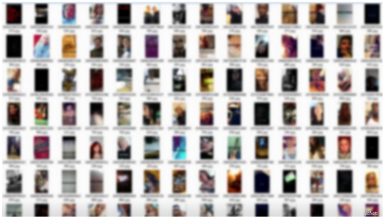
Progress brought complexity



Complexity caused bugs

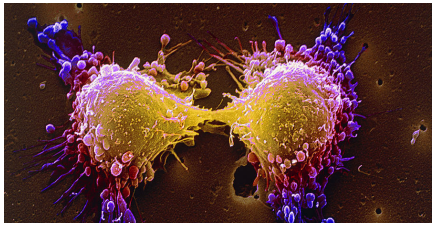


Blue screen



Leaked pictures

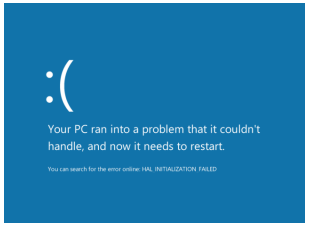
Dictators



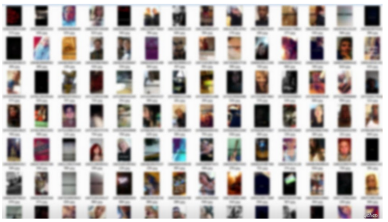
Cancer

Often our systems show undesired behaviors!!

Complexity caused bugs

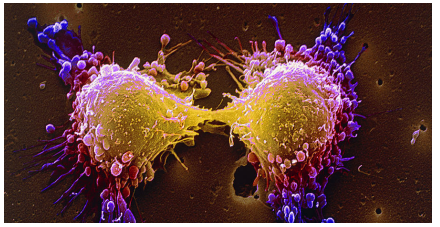


Blue screen



Leaked pictures

Dictators



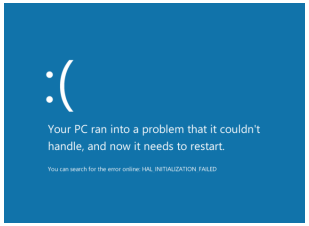
Cancer

Often our systems show undesired behaviors!!

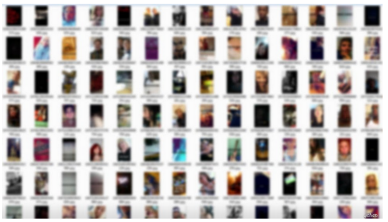
Why bugs happen?

How can we build bug-free systems?

Complexity caused bugs

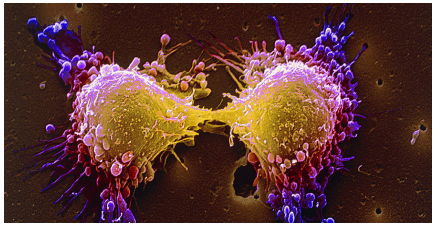


Blue screen



Leaked pictures

Dictators



Cancer

Often our systems show undesired behaviors!!

Why bugs happen?

How can we build bug-free systems?

Let us look at an example (play video).

Example: Ariane 5 failure

From the failure report: "... software exception (in the navigation system) was caused during execution of a data conversion from 64-bit floating point to 16-bit ... (the) number which was converted had a value greater than what could be represented by a 16-bit ..."

Example: Ariane 5 failure

From the failure report: "... software exception (in the navigation system) was caused during execution of a data conversion from 64-bit floating point to 16-bit ... (the) number which was converted had a value greater than what could be represented by a 16-bit ..."

"Although the source of the Operand Error has been identified, this in itself did not cause the mission to fail. The reason .... **lies in the culture within the Ariane programme** of only addressing random hardware failures."

Example: Ariane 5 failure

From the failure report: "... software exception (in the navigation system) was caused during execution of a data conversion from 64-bit floating point to 16-bit ... (the) number which was converted had a value greater than what could be represented by a 16-bit ..."

"Although the source of the Operand Error has been identified, this in itself did not cause the mission to fail. The reason .... **lies in the culture within the Ariane programme** of only addressing random hardware failures."

Culture = the belief that all errors have normal distributions

Example: Ariane 5 failure

From the failure report: "... software exception (in the navigation system) was caused during execution of a data conversion from 64-bit floating point to 16-bit ... (the) number which was converted had a value greater than what could be represented by a 16-bit ..."

"Although the source of the Operand Error has been identified, this in itself did not cause the mission to fail. The reason .... **lies in the culture within the Ariane programme** of only addressing random hardware failures."

Culture = the belief that all errors have normal distributions

The backup device also failed!! Software bugs have no nice distribution!

Example: Ariane 5 failure

From the failure report: "... software exception (in the navigation system) was caused during execution of a data conversion from 64-bit floating point to 16-bit ... (the) number which was converted had a value greater than what could be represented by a 16-bit ..."

"Although the source of the Operand Error has been identified, this in itself did not cause the mission to fail. The reason .... **lies in the culture within the Ariane programme** of only addressing random hardware failures."

Culture = the belief that all errors have normal distributions

The backup device also failed!! Software bugs have no nice distribution!

Essentially, a misunderstanding of the nature of the system.

Desired behaviors

First we need to ask,

what do we expect from our systems?

Desired behaviors

First we need to ask,

what do we expect from our systems?

For a system, we need to have goals that define the set of desired behaviors.

Desired behaviors

First we need to ask,

what do we expect from our systems?

For a system, we need to have goals that define the set of desired behaviors.

For example, a rocket should have the following goals

- it does not explode in flight (safety)
- it eventually reaches to the orbit (liveness)
- ...

Desired behaviors

First we need to ask,

what do we expect from our systems?

For a system, we need to have goals that define the set of desired behaviors.

For example, a rocket should have the following goals

- it does not explode in flight (safety)
- it eventually reaches to the orbit (liveness)
- ...

The specifications may not be explicitly available to us.

Safety vs. Liveness

Both goals are often in odds with each other.

Safety vs. Liveness

Both goals are often in odds with each other.

Liveness wants to move and safety wants to play conservative.

Safety vs. Liveness

Both goals are often in odds with each other.

Liveness wants to move and safety wants to play conservative.

Designing a system that is both safe and live is hard.

Safety vs. Liveness

Both goals are often in odds with each other.

Liveness wants to move and safety wants to play conservative.

Designing a system that is both safe and live is hard.

Example 1.1

Government

Safety

Liveness

Development tools

Once we have the goals then we need right set of tools to design the system

Development tools

Once we have the goals then we need right set of tools to design the system

For example,

- Programming language or instruments
- Organizational structure
- Skills of people

Example: bad excel

- In a 2010 paper, famous economists Reinhart and Rogoff inferred from the past century data that excessive debt hampers growth

Example: bad excel

- In a 2010 paper, famous economists Reinhart and Rogoff inferred from the past century data that excessive debt hampers growth
- The paper quickly became a classic for the austerity hawks

Example: bad excel

- In a 2010 paper, famous economists Reinhart and Rogoff inferred from the past century data that excessive debt hampers growth
- The paper quickly became a classic for the austerity hawks
- However, a student spotted a problem. Their spreadsheet skipped key data points, which biased the results in favour of the inference.

Example: bad excel

- In a 2010 paper, famous economists Reinhart and Rogoff inferred from the past century data that excessive debt hampers growth
- The paper quickly became a classic for the austerity hawks
- However, a student spotted a problem. Their spreadsheet skipped key data points, which biased the results in favour of the inference.

	B	C	I	J	K	L	M
2			Real GDP growth				
3			Debt/GDP				
4	Country	Coverage	30 or less	30 to 60	60 to 90	90 or above	30 or less
26			3.7	3.0	3.5	1.7	5.5
27	Minimum		1.6	0.3	1.3	-1.8	0.8
28	Maximum		5.4	4.9	10.2	3.6	13.3
29							
30	US	1946-2009	n.a.	3.4	3.3	-2.0	n.a.
31	UK	1946-2009	n.a.	2.4	2.5	2.4	n.a.
32	Sweden	1946-2009	3.6	2.9	2.7	n.a.	6.3
33	Spain	1946-2009	1.5	3.4	4.2	n.a.	9.9
34	Portugal	1952-2009	4.8	2.5	0.3	n.a.	7.9
35	New Zealand	1948-2009	2.5	2.9	3.9	-7.9	2.6
36	Netherlands	1956-2009	4.1	2.7	1.1	n.a.	6.4
37	Norway	1947-2009	3.4	5.1	n.a.	n.a.	5.4
38	Japan	1946-2009	7.0	4.0	1.0	0.7	7.0
39	Italy	1951-2009	5.4	2.1	1.8	1.0	5.6
40	Ireland	1948-2009	4.4	4.5	4.0	2.4	2.9
41	Greece	1970-2009	4.0	0.3	2.7	2.9	13.3
42	Germany	1946-2009	3.9	0.9	n.a.	n.a.	3.2
43	France	1949-2009	4.9	2.7	3.0	n.a.	5.2
44	Finland	1946-2009	3.8	2.4	5.5	n.a.	7.0
45	Denmark	1950-2009	3.5	1.7	2.4	n.a.	5.6
46	Canada	1951-2009	1.9	3.6	4.1	n.a.	2.2
47	Belgium	1947-2009	n.a.	4.2	3.1	2.6	n.a.
48	Austria	1948-2009	5.2	3.3	-3.8	n.a.	5.7
49	Australia	1951-2009	3.2	4.9	4.0	n.a.	5.9
50							
51			4.1	2.8	2.8	=AVERAGE(L30:L44)	

Last five rows are skipped!

Example: bad excel (contd.)

The key point is that excel is a bad programming environment

Example: bad excel (contd.)

The key point is that excel is a bad programming environment

- the program is not visible to the user - one only sees cells with numbers

Example: bad excel (contd.)

The key point is that excel is a bad programming environment

- the program is not visible to the user - one only sees cells with numbers
- data and program are not separated

Example: bad excel (contd.)

The key point is that excel is a bad programming environment

- the program is not visible to the user - one only sees cells with numbers
- data and program are not separated
- no debugging tool

Example: bad excel (contd.)

The key point is that excel is a bad programming environment

- the program is not visible to the user - one only sees cells with numbers
- data and program are not separated
- no debugging tool
- hard to monitor changes

Example: bad excel (contd.)

The key point is that excel is a bad programming environment

- the program is not visible to the user - one only sees cells with numbers
- data and program are not separated
- no debugging tool
- hard to monitor changes
- almost impossibility of code review

Example: bad excel (contd.)

The key point is that excel is a bad programming environment

- the program is not visible to the user - one only sees cells with numbers
- data and program are not separated
- no debugging tool
- hard to monitor changes
- almost impossibility of code review

Excel should not be used for any serious work!!

Tools for analysis

Once we have built the system, we need an appropriate analysis method to check that the system stisfies with the goals.

Tools for analysis

Once we have built the system, we need an appropriate analysis method to check that the system stisfies with the goals.

Example 1.2

All behaviors of the Ariane 5 software should have been analyzed.

Tools for analysis

Once we have built the system, we need an appropriate analysis method to check that the system stisfies with the goals.

Example 1.2

All behaviors of the Ariane 5 software should have been analyzed.

The software on such machines have more states then stars in the universe.

Tools for analysis

Once we have built the system, we need an appropriate analysis method to check that the system stisfies with the goals.

Example 1.2

All behaviors of the Ariane 5 software should have been analyzed.

The software on such machines have more states then stars in the universe.

In the state space, the distribution of the errors is unknown.

Tools for analysis

Once we have built the system, we need an appropriate analysis method to check that the system stisfies with the goals.

Example 1.2

All behaviors of the Ariane 5 software should have been analyzed.

The software on such machines have more states then stars in the universe.

In the state space, the distribution of the errors is unknown.

We are dealing with an ugly beast!

Only continuous math is inappropriate

The classic methods such as differential equations, linear optimizations, simulation, important sampling, etc are the shiny knights that have slayed many problems.

Unfortunately, these methods are insufficient in our setting.

Bugs hunting needs combinatorial reasoning!!

It is time to get rough! Nice approximations do not work.

Bugs hunting needs combinatorial reasoning!!

It is time to get rough! Nice approximations do not work.

We have to search the combinatorial space for the analysis.

Bugs hunting needs combinatorial reasoning!!

It is time to get rough! Nice approximations do not work.

We have to search the combinatorial space for the analysis.

We need to make unholy alliances. The search is often aided by smart optimizations and machine learning that tells where to search first.

Topic 1.1

Formal Verification 101

“Fronts” of software reliability

Language design

Systematic testing

“Fronts” of software reliability

Language design | Systematic testing | Programming environments | Technical education

"Fronts" of software reliability

Language design | Systematic testing | Programming environments | Technical education

"Fronts" of software reliability

Formal Verification | Automated Synthesis

Language design | Systematic testing | Programming environments | Technical education

"Fronts" of software reliability

Formal Verification | Automated Synthesis

Heterogeneous technologies need to work together for effective reliability

Verification problem

$$\text{Program} \overset{?}{\Rightarrow} \text{Property}$$

- Hardware
- Software

- Safety
- Liveness
- Quantitative
- Probabilistic

Software verification

- Desired property is expresses as a logical formula ψ
- For a given program P, we aim to prove theorem

$$P \Rightarrow \psi$$

(all behaviors of P satisfy ψ)

Abstraction is the key method to prove $P \Rightarrow \psi$

Abstraction is the key method to prove $P \Rightarrow \psi$

- P moves from a state to another state

Abstraction is the key method to prove $P \Rightarrow \psi$

- P moves from a state to another state
- Abstract model $P^{\#}$ moves from
  a set of states to another set of states
  (abstract model has more behaviors than program)

Abstraction is the key method to prove $P \Rightarrow \psi$

- P moves from a state to another state
- Abstract model $P^{\#}$ moves from
  a set of states to another set of states
  (abstract model has more behaviors than program)
- We prove the following implications

$$P \Rightarrow P^{\#} \Rightarrow \psi$$

Abstraction is the key method to prove $P \Rightarrow \psi$

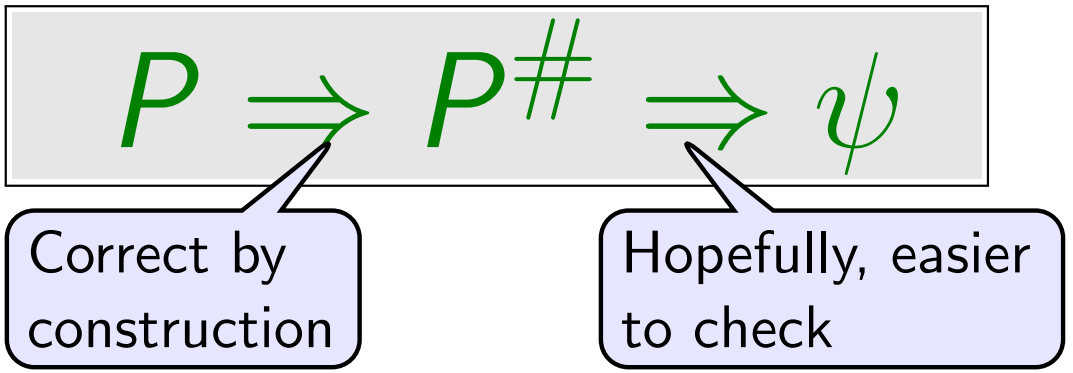
- P moves from a state to another state
- Abstract model $P^{\#}$ moves from
  a set of states to another set of states
  (abstract model has more behaviors than program)
- We prove the following implications

$$P \Rightarrow P^{\#} \Rightarrow \psi$$

Correct by construction

Abstraction is the key method to prove $P \Rightarrow \psi$

- P moves from a state to another state
- Abstract model $P^{\#}$ moves from
  a set of states to another set of states
  (abstract model has more behaviors than program)
- We prove the following implications

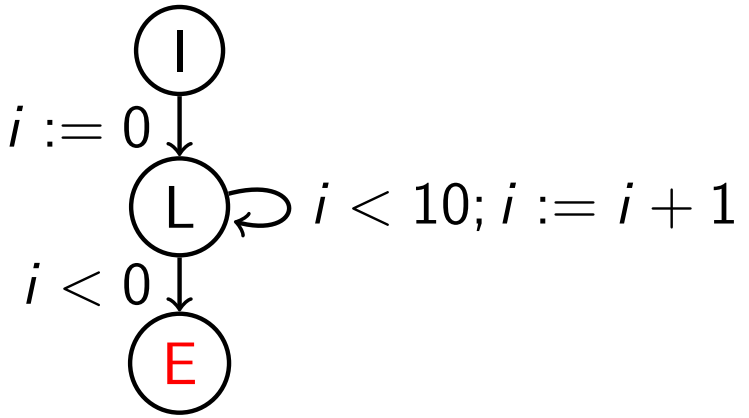


Example : program to CFG

```
void main() {
 i = 0;
 while( i < 10 ) {
   i++;
 }
 assert( i >= 0 );
}
```

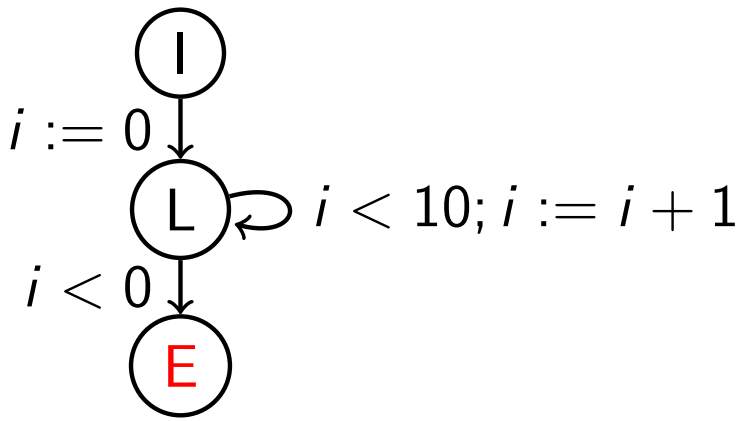
Example : program to CFG

```
void main() {
 i = 0;
 while( i < 10 ) {
   i++;
 }
 assert( i >= 0 );
}
```

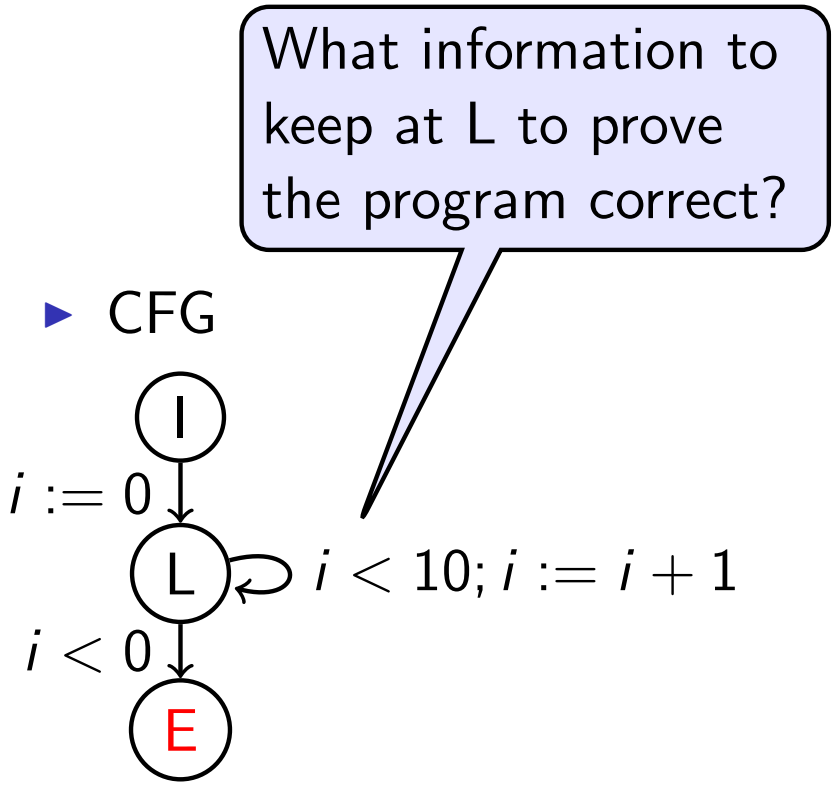


Example : CFG to abstract model

- CFG



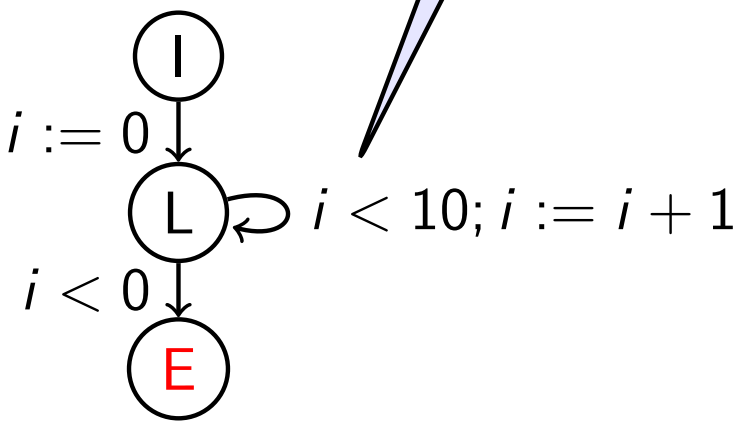
Example : CFG to abstract model



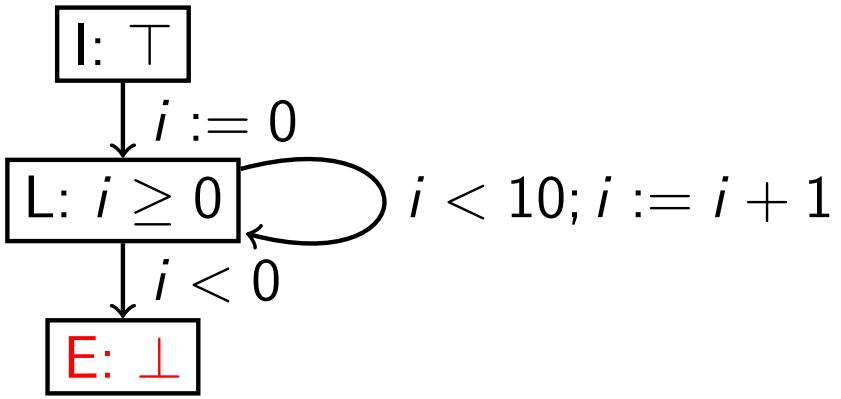
Example : CFG to abstract model

What information to keep at L to prove the program correct?

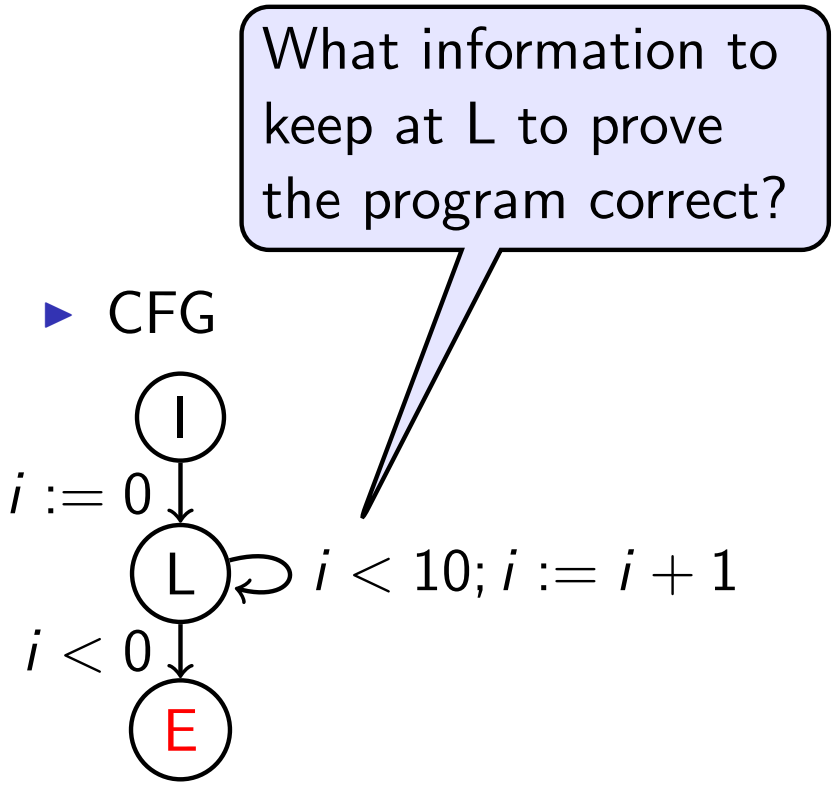
- CFG

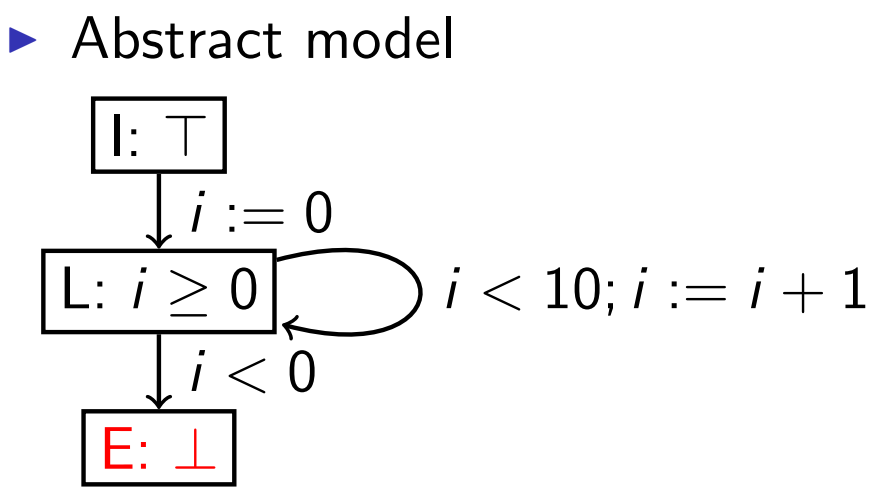


- Abstract model



Example : CFG to abstract model



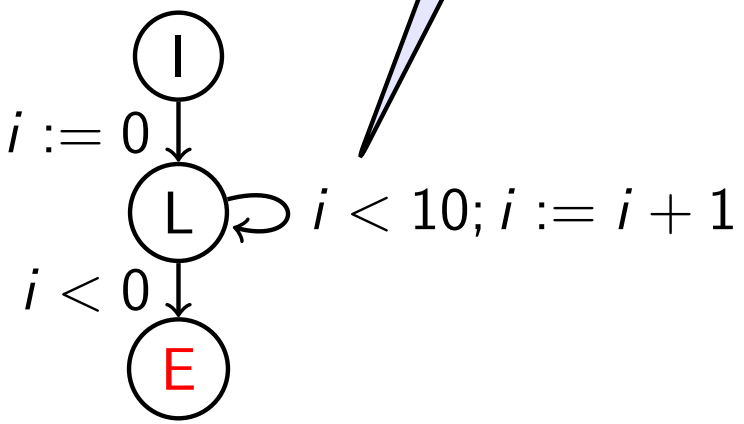


Verification problem $\equiv$ **find the right abstract model**

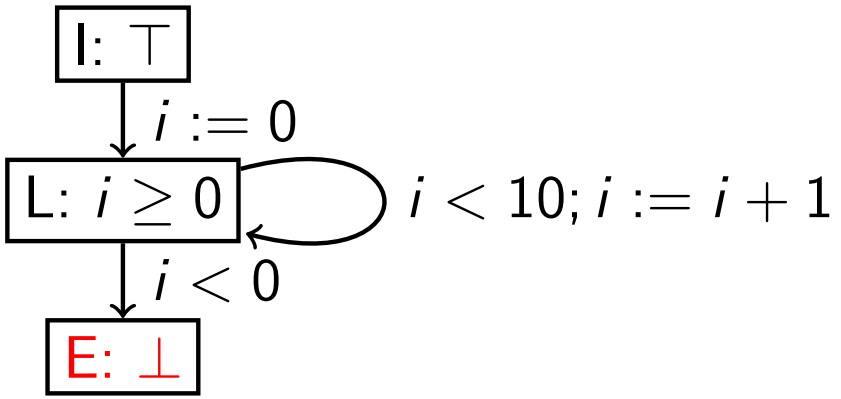
Example : CFG to abstract model

What information to keep at L to prove the program correct?

- CFG



- Abstract model



Verification problem $\equiv$ **find the right abstract model**

Verification methods only differ in how to find such an abstract model

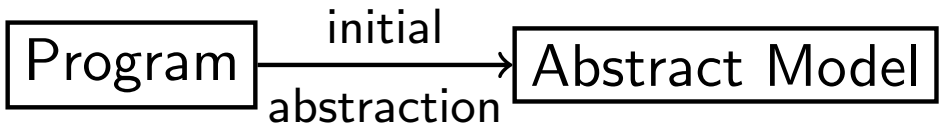
Example : A verification method

CEGAR: CounterExample Guided Abstraction Refinement

Program

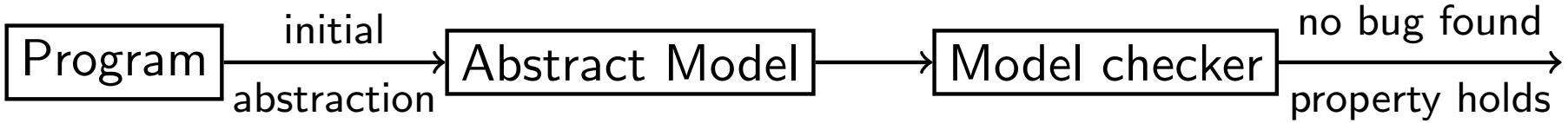
Example : A verification method

CEGAR: CounterExample Guided Abstraction Refinement



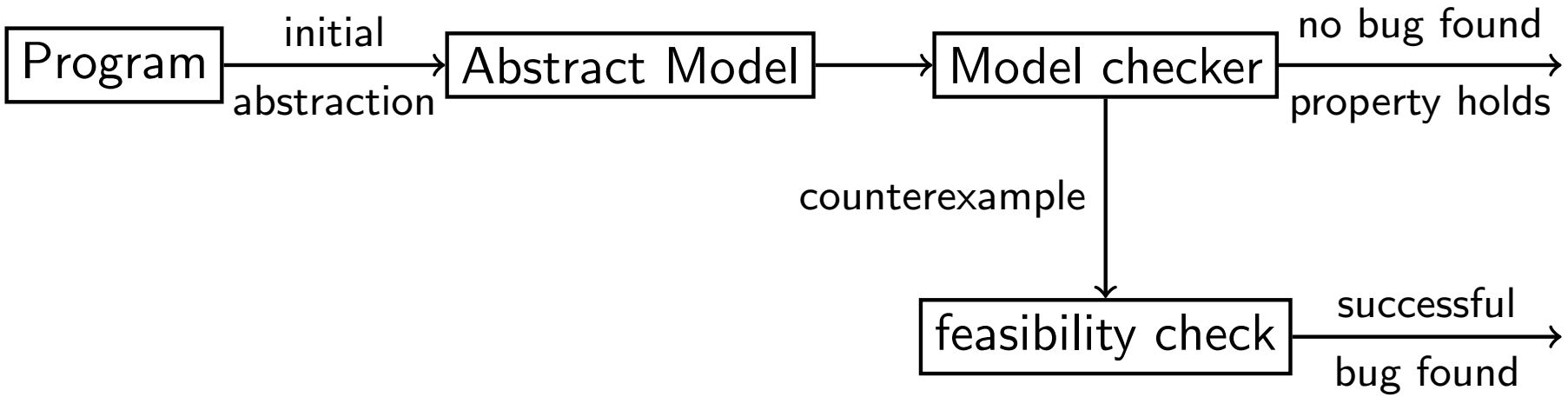
Example : A verification method

CEGAR: CounterExample Guided Abstraction Refinement



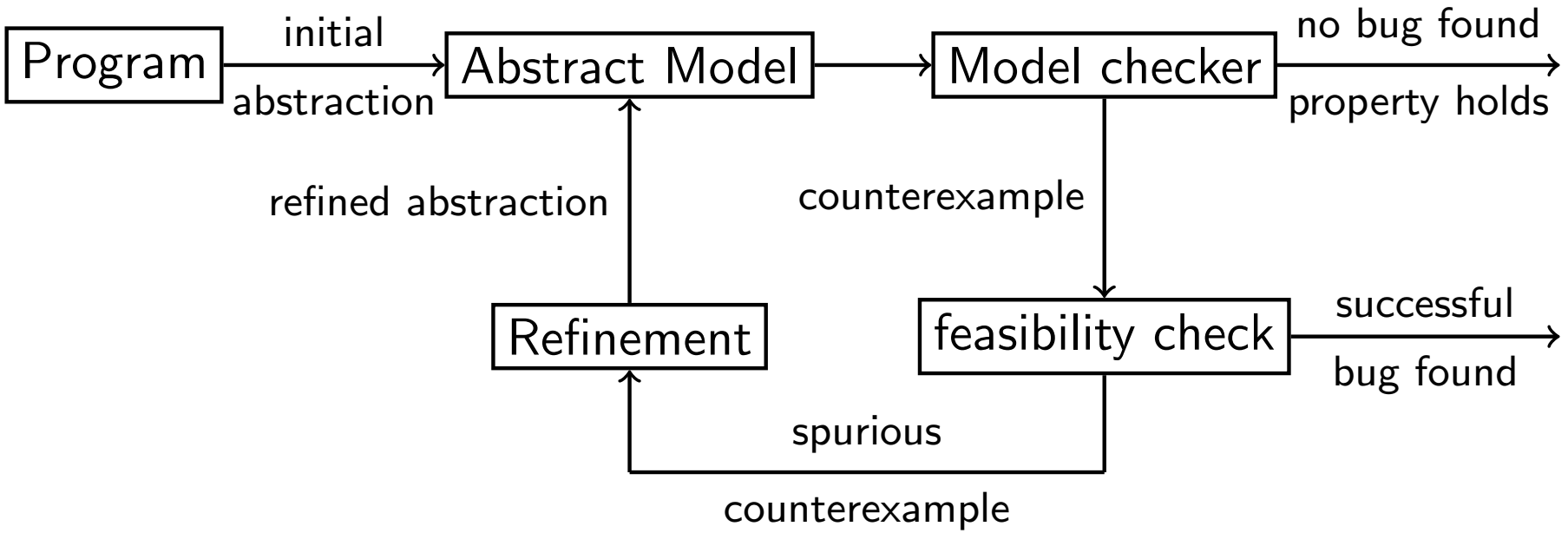
Example : A verification method

CEGAR: CounterExample Guided Abstraction Refinement



Example : A verification method

CEGAR: CounterExample Guided Abstraction Refinement



Topic 1.2

What is so hard about concurrency?

Schedule blowup

Exercise 1.1

What is the number of schedules between two threads with number of instructions N_1 and N_2 ?

Schedule blowup

Exercise 1.1

What is the number of schedules between two threads with number of instructions N_1 and N_2 ?

The blowup is not the only problem.

In the presence of synchronization primitives, the sets of allowed schedules appear deceptively simple, but are ugly beasts
e. g., locks, barriers, etc

Memory behavior

We usually believe that memory is sequential consistent.

Memory behavior

We usually believe that memory is sequential consistent.

In the concurrent world, threads may not have same view of memory!

Memory behavior

We usually believe that memory is sequential consistent.

In the concurrent world, threads may not have same view of memory!

Example 1.3

```
Global init result = 0, ready = 0;
```

Memory behavior

We usually believe that memory is sequential consistent.

In the concurrent world, threads may not have same view of memory!

Example 1.3

```
Global init result = 0, ready = 0;
```

Backend Thread

```
r = calculate();
result = r;
ready = 1;
```

Memory behavior

We usually believe that memory is sequential consistent.

In the concurrent world, threads may not have same view of memory!

Example 1.3

```
Global init result = 0, ready = 0;
```

Backend Thread		**Display Thread**
`r = calculate();`	‖	`while(ready == 0);`
`result = r;`		`print results;`
`ready = 1;`		

Memory behavior

We usually believe that memory is sequential consistent.

In the concurrent world, threads may not have same view of memory!

Example 1.3

```
Global init result = 0, ready = 0;
```

Backend Thread		**Display Thread**
`r = calculate();`	$\parallel$	`while(ready == 0);`
`result = r;`		`print results;`
`ready = 1;`		

Will this program always print the result of the calculation?

Memory behavior

We usually believe that memory is sequential consistent.

In the concurrent world, threads may not have same view of memory!

Example 1.3

```
Global init result = 0, ready = 0;
```

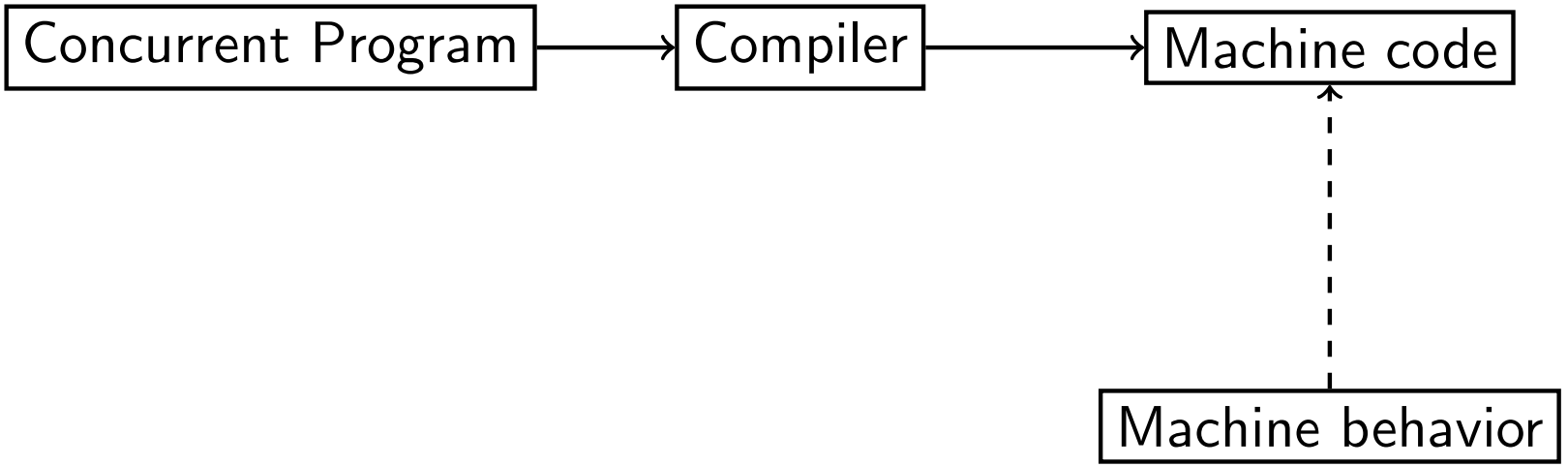
Backend Thread		**Display Thread**
`r = calculate();`	‖	`while(ready == 0);`
`result = r;`		`print results;`
`ready = 1;`		

Will this program always print the result of the calculation?

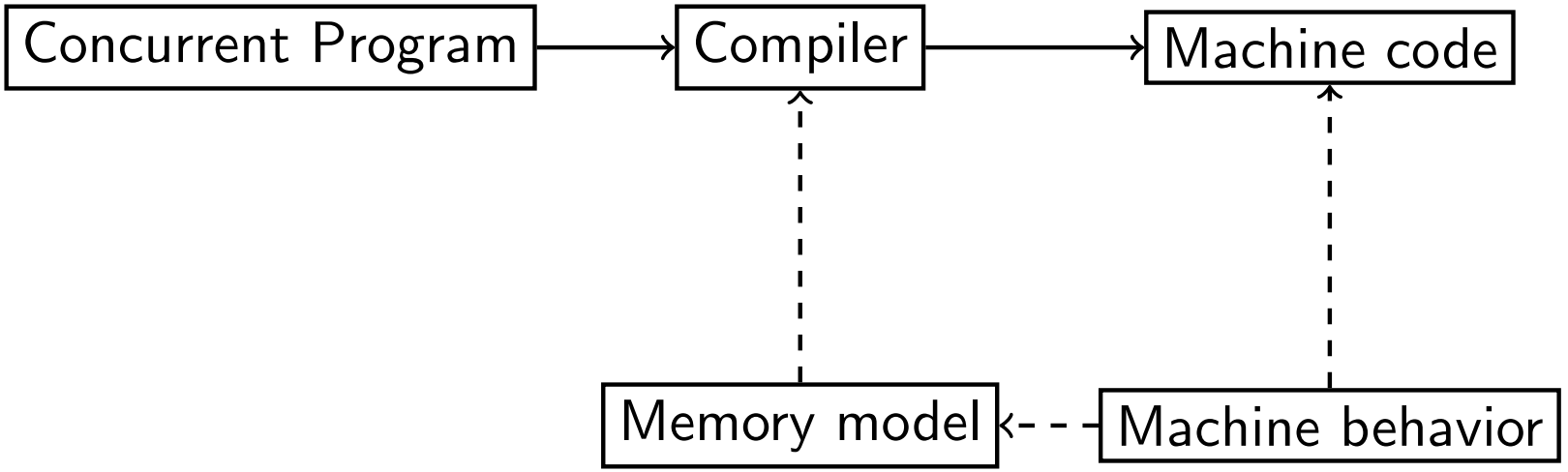
Writes overtake each other. The program is wrong on a typical smart phone!!

Memory models

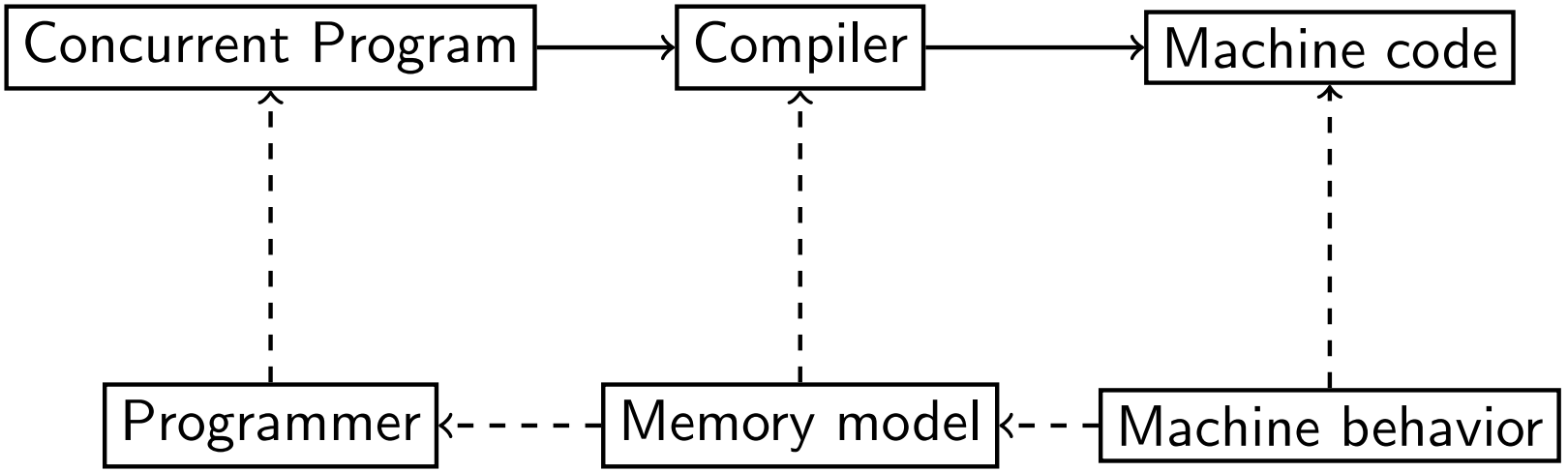
Memory models



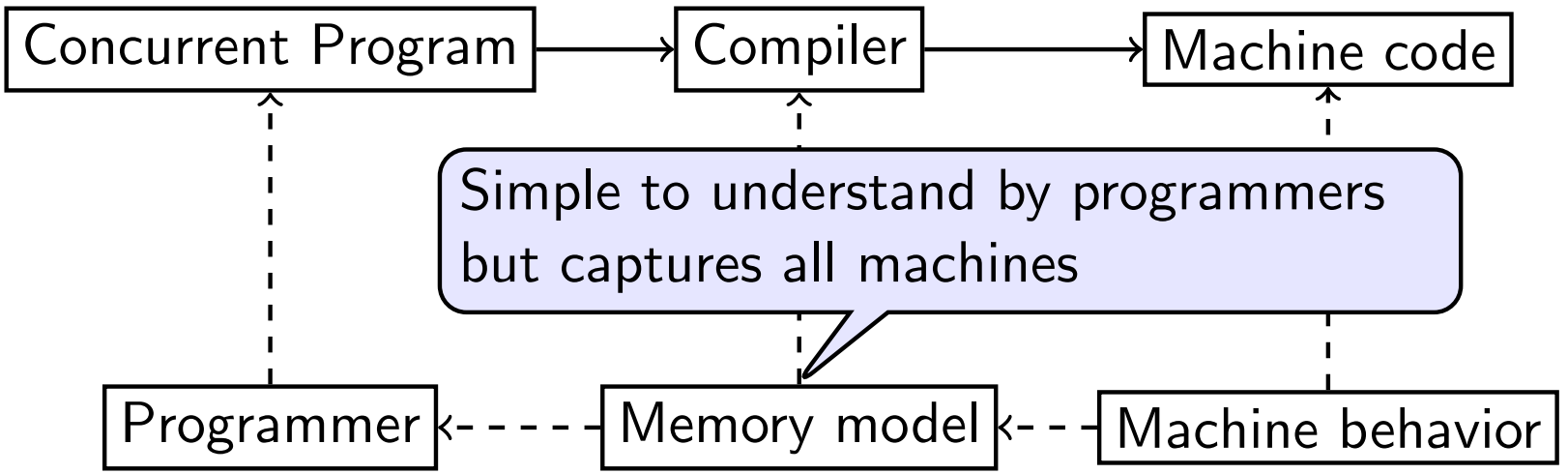
Memory models



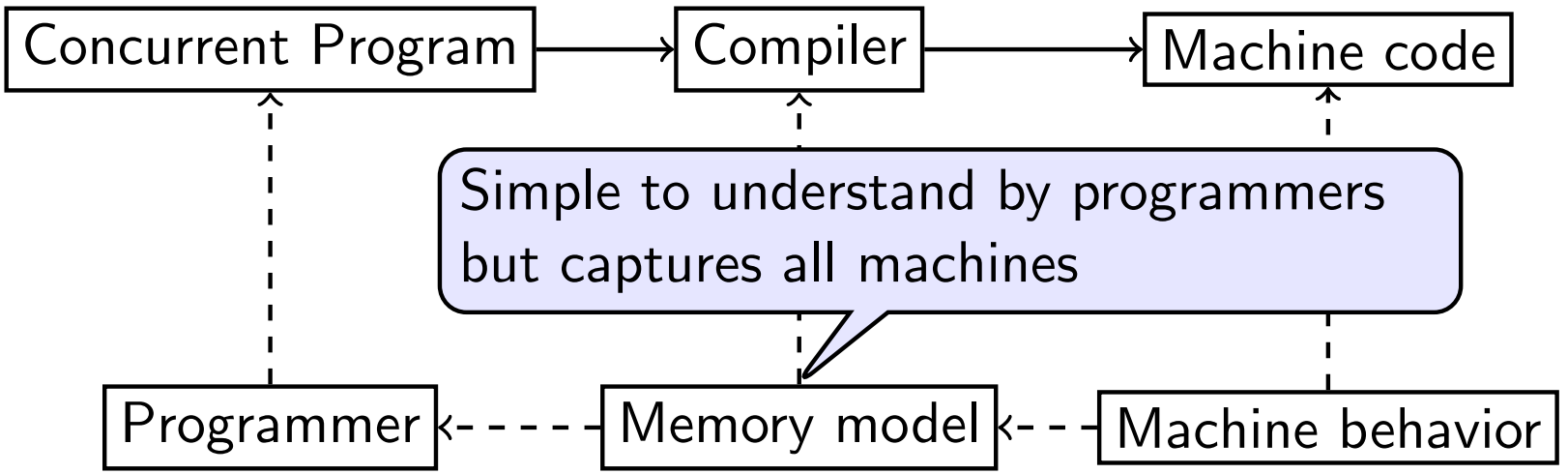
Memory models



Memory models

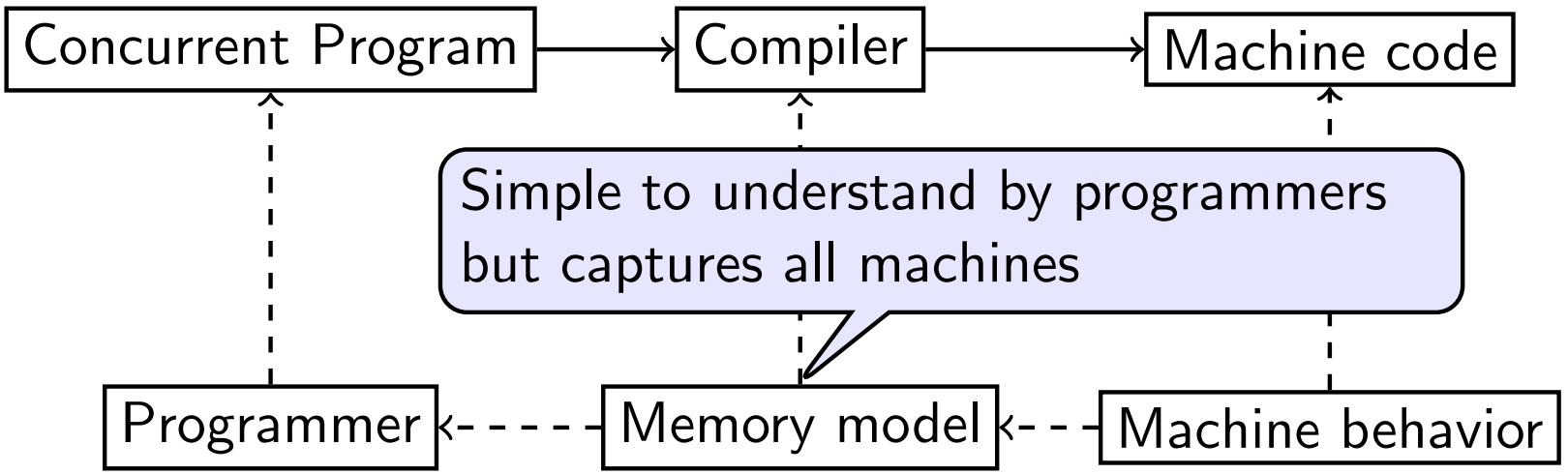


Memory models



- C++11 publishes such a memory model
- Allows too many behaviors, even if no hardware exhibits them
- Disallows many simple compiler optimizations

Memory models



- C++11 publishes such a memory model
- Allows too many behaviors, even if no hardware exhibits them
- Disallows many simple compiler optimizations

- We are working on developing memory models that
  - is easy to understand
  - allows a good number of compiler optimization
  - allows efficient analysis of programs