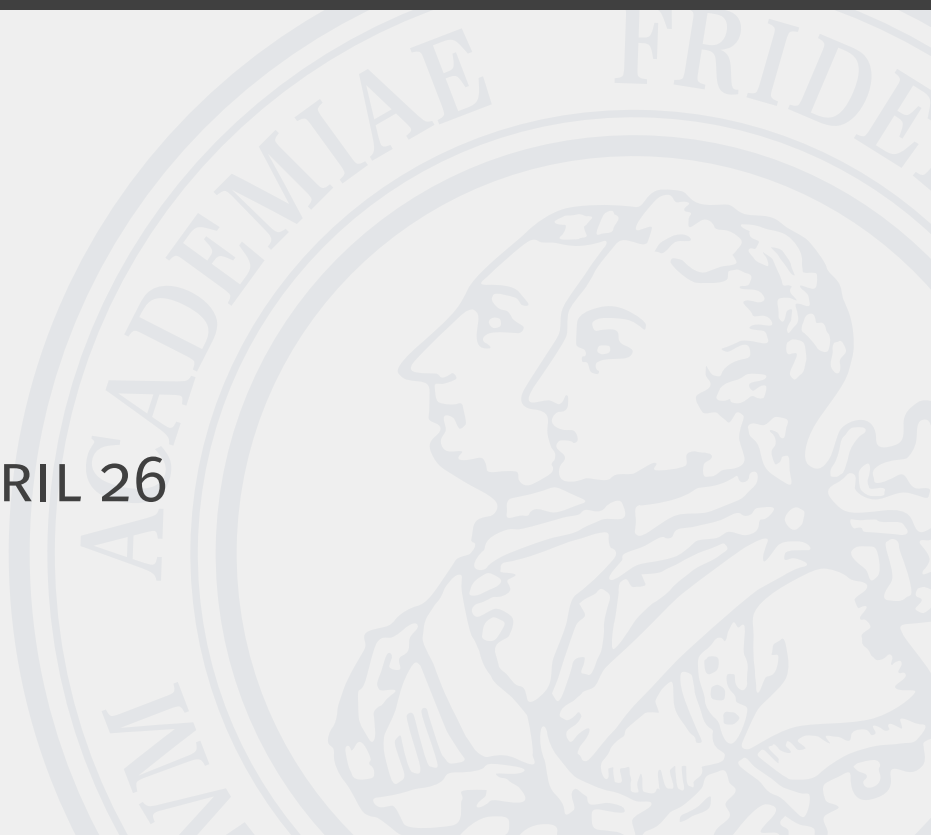
Unifying Notions of Feedback

Sergey Goncharov



FAU Tag der Informatik 2019, April 26

Unifying Notions of Feedback

Sergey Goncharov

FAU Tag der Informatik 2019, April 26

Outline

- Semantics of programs and specification frameworks is a rich ecosystem containing a tremendous bulk of methods and tools for dealing with various flavours of computation from classical, nondeterministic, probabilistic to quantum

Outline

- Semantics of programs and specification frameworks is a rich ecosystem containing a tremendous bulk of methods and tools for dealing with various flavours of computation from classical, nondeterministic, probabilistic to quantum
- A unifying language for semantics is category theory

Outline

- Semantics of programs and specification frameworks is a rich ecosystem containing a tremendous bulk of methods and tools for dealing with various flavours of computation from classical, nondeterministic, probabilistic to quantum
- A unifying language for semantics is category theory
- Feedback is a distinctive feature of complex systems, computationally interpreted e.g. as iteration or recursion

Outline

- Semantics of programs and specification frameworks is a rich ecosystem containing a tremendous bulk of methods and tools for dealing with various flavours of computation from classical, nondeterministic, probabilistic to quantum
- A unifying language for semantics is category theory
- Feedback is a distinctive feature of complex systems, computationally interpreted e.g. as iteration or recursion
- Total (but not partial!) feedback operators are categorically unified with traced (monoidal) categories

Outline

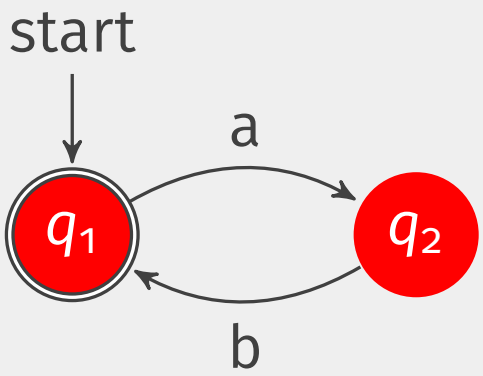
- Semantics of programs and specification frameworks is a rich ecosystem containing a tremendous bulk of methods and tools for dealing with various flavours of computation from classical, nondeterministic, probabilistic to quantum
- A unifying language for semantics is category theory
- Feedback is a distinctive feature of complex systems, computationally interpreted e.g. as iteration or recursion
- Total (but not partial!) feedback operators are categorically unified with traced (monoidal) categories
- Grand unification:

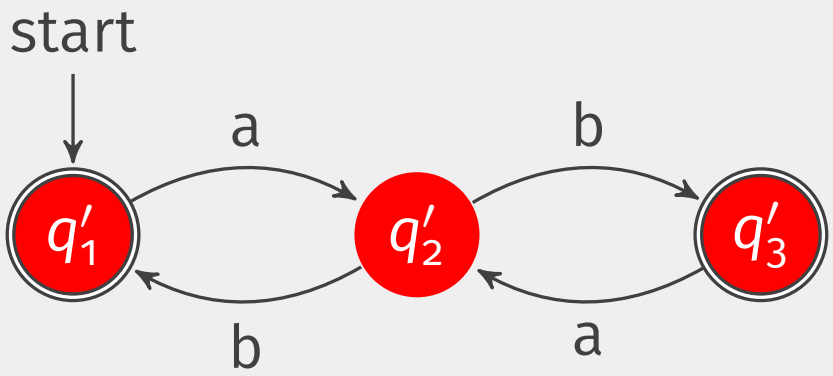
Guarded Traced Categories

Why Semantics?

A Well-known Scenario

How do we know that automata

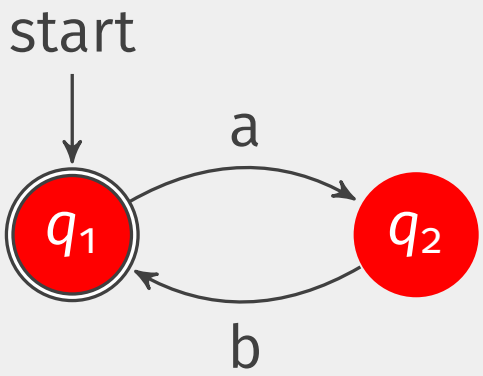


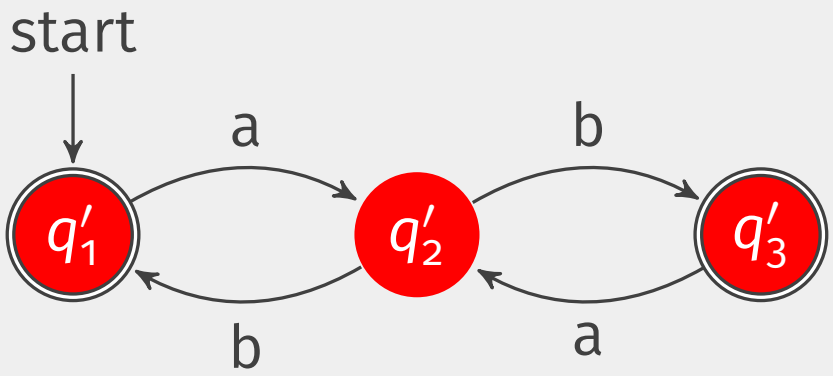


are equivalent?

A Well-known Scenario

How do we know that automata





are equivalent?

dinaturality identity

Because $[\![q_1]\!] = (ab)^\star$, $[\![q'_1]\!] = a(ba)^\star b + 1$ and: $(ab)^\star = a(ba)^\star b + 1$

semantic brackets

Semantics in Computer Science

Semantics

Semantics in Computer Science

Effectiveness
(of presentation, decidability)

Semantics

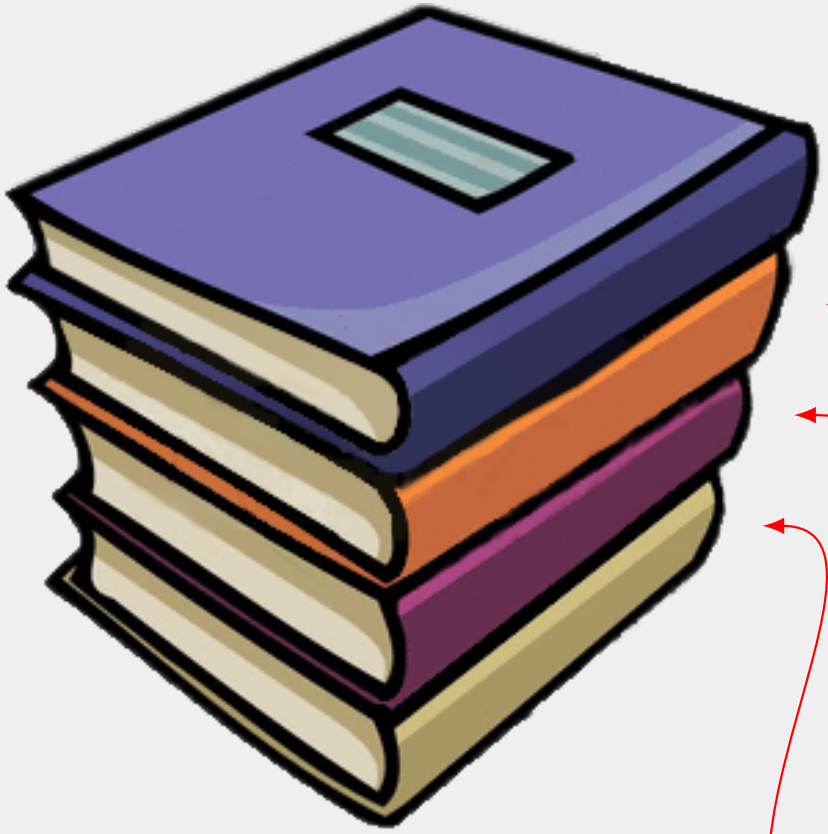
Semantics in Computer Science

Efficiency (complexity of computations, efficient data structures)

Effectiveness (of presentation, decidability)

Semantics

Semantics in Computer Science



Algorithms, implementations, concrete programming languages, compilers, interpreters, simulators, proof assistants, ...

Efficiency (complexity of computations, efficient data structures)

Effectiveness (of presentation, decidability)

Semantics

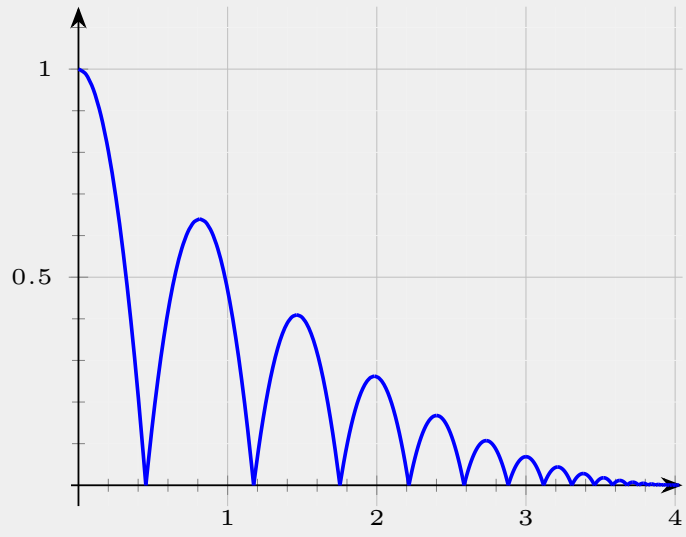
A Less Known Scenario

Bouncing ball is a simple Newtonian system specified by differential equation $\ddot{h} = -g$ ($g \approx 9.8$) whose solution is

$$h(t) = h_0 + v_0 t - \frac{gt^2}{2}$$

with initial values:

- $v_0 = 0, h_0 \neq 0$ (peak height)
- $h_0 = 0, v_0 \neq 0$ (zero height)



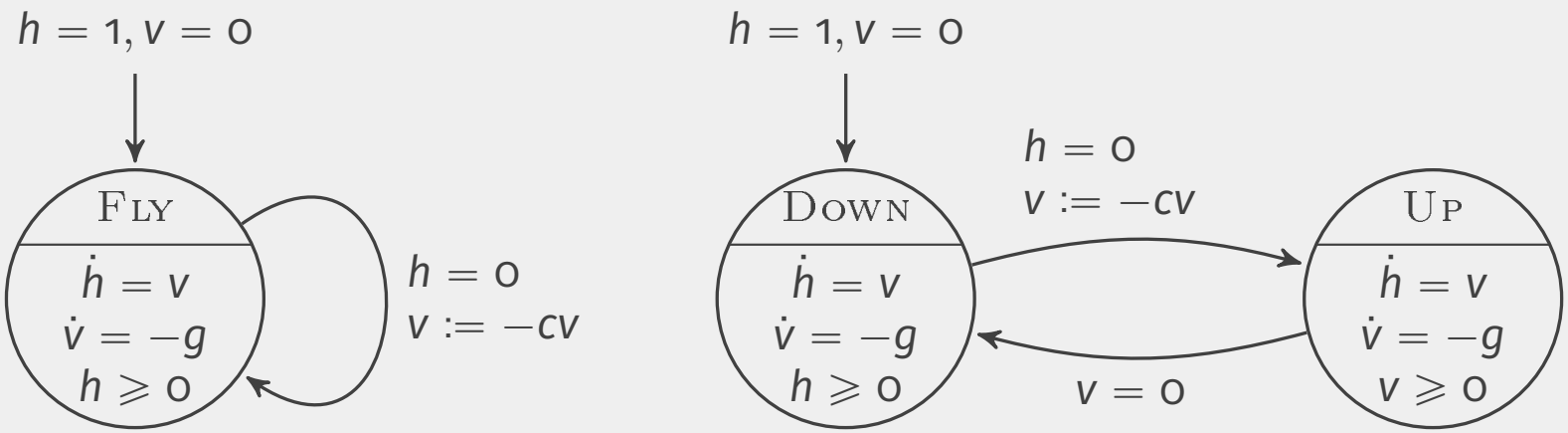
Features:

- deterministic
- hybrid: the velocity changes discretely at the bottom $v \mapsto -cv$, but it changes continuously in the meanwhile
- Zeno behaviour: the state of rest is only reachable in the limit

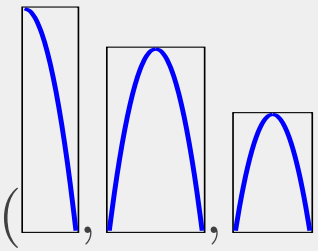
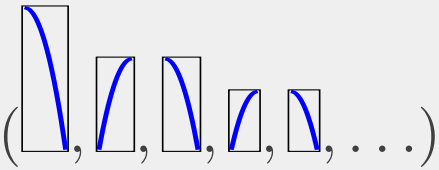
damping factor

Hybrid Automata

The following hybrid automata "A" and "B" capture the bouncing ball behaviour:



These automata are not equivalent under standard semantics, because

$[\![A]\!] = ($  $, \ldots)$, but $[\![B]\!] = ($  $, \ldots)$

Impact of Semantics

- Knowing the semantics of automata, we can minimize them, transform, prove equivalence

Impact of Semantics

- Knowing the semantics of automata, we can minimize them, transform, prove equivalence
- We can transfer knowledge between different models, as the theories of nondeterministic, probablilistic, push-down, etc, etc automata have a lot in common

Impact of Semantics

- Knowing the semantics of automata, we can minimize them, transform, prove equivalence
- We can transfer knowledge between different models, as the theories of nondeterministic, probablilistic, push-down, etc, etc automata have a lot in common
- We can optimize programs, e.g.

```
while b do ...                    while b do ...
  if b then  ...          →         ...
  else /* dead code */            done
done
```

and verify them (since, we know what they mean!)

Impact of Semantics

- Knowing the semantics of automata, we can minimize them, transform, prove equivalence
- We can transfer knowledge between different models, as the theories of nondeterministic, probablilistic, push-down, etc, etc automata have a lot in common
- We can optimize programs, e.g.

```
while b do ...                    while b do ...
  if b then  ...          →         ...
  else /* dead code */            done
done
```

  and verify them (since, we know what they mean!)
- Principled semantic foundations improve design of languages, software and hardware systems (types, compositionality, Curry-Howard correspondence, etc)

Why Category Theory?

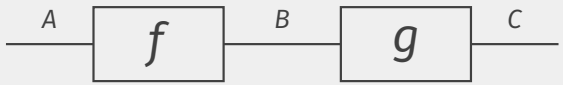
[...] Kategorientheorie – ein sehr komplexes Gebiet mit tiefen mathematischen Wurzeln, und mit relativ wenigen Experten auf diesem Gebiet

— Anonymous referee

Categories: Quick Intro

- A category $\mathbf{C}$ consists of wires (=objects) $|\mathbf{C}|$ and boxes (=morphisms) $\mathbf{C}(A, B)$ with $A, B \in |\mathbf{C}|$, which can be combined:

identity wire

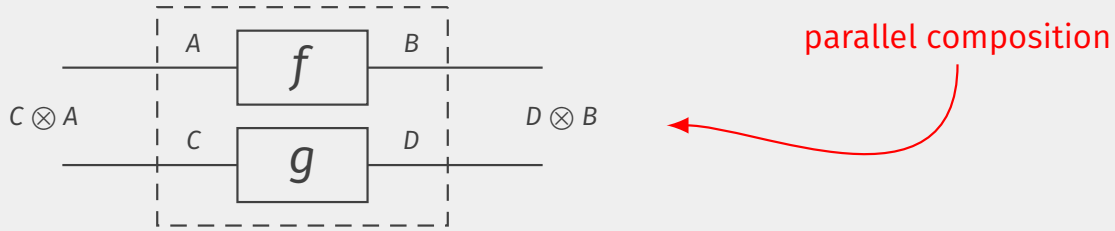


sequential composition

Categories: Quick Intro

- A category $\mathbf{C}$ consists of wires (=objects) $|\mathbf{C}|$ and boxes (=morphisms) $\mathbf{C}(A, B)$ with $A, B \in |\mathbf{C}|$, which can be combined:

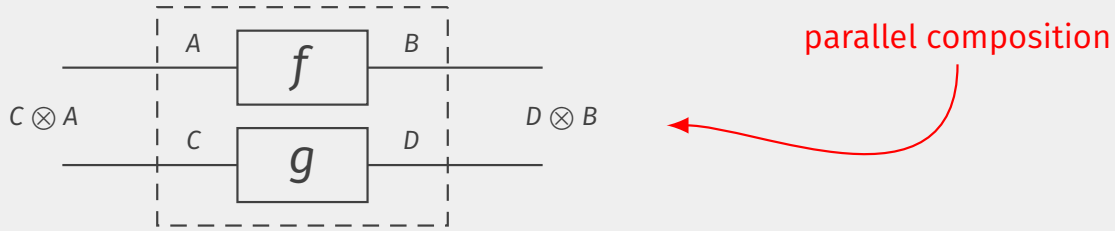
- A category is monoidal if morphisms can be tensored:



Categories: Quick Intro

- A category $\mathbf{C}$ consists of wires (=objects) $|\mathbf{C}|$ and boxes (=morphisms) $\mathbf{C}(A, B)$ with $A, B \in |\mathbf{C}|$, which can be combined:

- A category is monoidal if morphisms can be tensored:

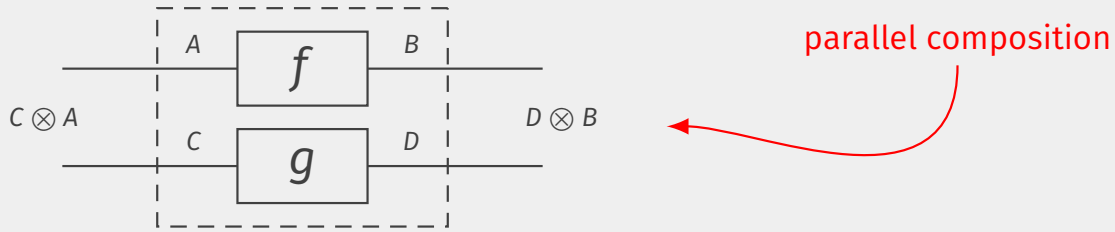


- A monoidal category is symmetric if wires can be crossed:

Categories: Quick Intro

- A category $\mathbf{C}$ consists of wires (=objects) $|\mathbf{C}|$ and boxes (=morphisms) $\mathbf{C}(A, B)$ with $A, B \in |\mathbf{C}|$, which can be combined:

- A category is monoidal if morphisms can be tensored:

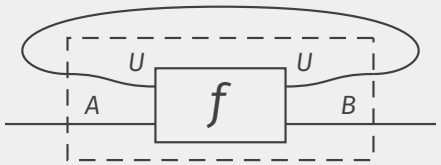


- A monoidal category is symmetric if wires can be crossed:

Boxes are intuitively: programs, processes, components, automata; wires are types, communication channels

Traced Categories

Traced categories additionally allow feedback loops, called traces:



Traced categories provide a unifying framework for

- Iteration (roughly: while-loops)
- Recursion (roughly: fixpoint combinators of λ-calculus)
- Knot theory
- Operator theory (e.g. traces model quantum measurements)

Why Guarded Traces?

Automata, Revisited

Consider again the regular expressions $(ab)^\star$ and $a(ba)^\star b + 1$
Here, Kleene star $e^\star$ is the unique fixpoint of

$$x \mapsto ex + 1$$

Equation $(ab)^\star = a(ba)^\star b + 1$ is true, because $a(ba)^\star b + 1$ is a fixpoint of the same map

Automata, Revisited

Consider again the regular expressions $(ab)^\star$ and $a(ba)^\star b + 1$
Here, Kleene star $e^\star$ is the unique fixpoint of

$$x \mapsto ex + 1$$

Equation $(ab)^\star = a(ba)^\star b + 1$ is true, because $a(ba)^\star b + 1$ is a fixpoint of the same map:

$$a(ba)^\star b + 1 = a((ba)(ba)^\star + 1)b + 1$$

Automata, Revisited

Consider again the regular expressions $(ab)^\star$ and $a(ba)^\star b + 1$
Here, Kleene star $e^\star$ is the unique fixpoint of

$$x \mapsto ex + 1$$

Equation $(ab)^\star = a(ba)^\star b + 1$ is true, because $a(ba)^\star b + 1$ is a fixpoint of the same map:

$$\begin{aligned} a(ba)^\star b + 1 &= a((ba)(ba)^\star + 1)b + 1 \\ &= a(ba)(ba)^\star b + a1b + 1 \end{aligned}$$

Automata, Revisited

Consider again the regular expressions $(ab)^\star$ and $a(ba)^\star b + 1$
Here, Kleene star $e^\star$ is the unique fixpoint of

$$x \mapsto ex + 1$$

Equation $(ab)^\star = a(ba)^\star b + 1$ is true, because $a(ba)^\star b + 1$ is a fixpoint of the same map:

$$\begin{aligned} a(ba)^\star b + 1 &= a((ba)(ba)^\star + 1)b + 1 \\ &= a(ba)(ba)^\star b + a1b + 1 \\ &= (ab)a(ba)^\star b + ab + 1 \end{aligned}$$

Automata, Revisited

Consider again the regular expressions $(ab)^\star$ and $a(ba)^\star b + 1$
Here, Kleene star $e^\star$ is the unique fixpoint of

$$x \mapsto ex + 1$$

Equation $(ab)^\star = a(ba)^\star b + 1$ is true, because $a(ba)^\star b + 1$ is a fixpoint of the same map:

$$\begin{aligned} a(ba)^\star b + 1 &= a((ba)(ba)^\star + 1)b + 1 \\ &= a(ba)(ba)^\star b + a1b + 1 \\ &= (ab)a(ba)^\star b + ab + 1 \\ &= (ab)(a(ba)^\star b + 1) + 1 \end{aligned}$$

Automata, Revisited

Consider again the regular expressions $(ab)^\star$ and $a(ba)^\star b + 1$
Here, Kleene star $e^\star$ is the unique fixpoint of

$$x \mapsto ex + 1$$

Equation $(ab)^\star = a(ba)^\star b + 1$ is true, because $a(ba)^\star b + 1$ is a fixpoint of the same map:

$$\begin{aligned} a(ba)^\star b + 1 &= a((ba)(ba)^\star + 1)b + 1 \\ &= a(ba)(ba)^\star b + a1b + 1 \\ &= (ab)a(ba)^\star b + ab + 1 \\ &= (ab)(a(ba)^\star b + 1) + 1 \end{aligned}$$

Automata, Revisited

Consider again the regular expressions $(ab)^\star$ and $a(ba)^\star b + 1$
Here, Kleene star $e^\star$ is the unique fixpoint of

$$x \mapsto ex + 1$$

Equation $(ab)^\star = a(ba)^\star b + 1$ is true, because $a(ba)^\star b + 1$ is a fixpoint of the same map:

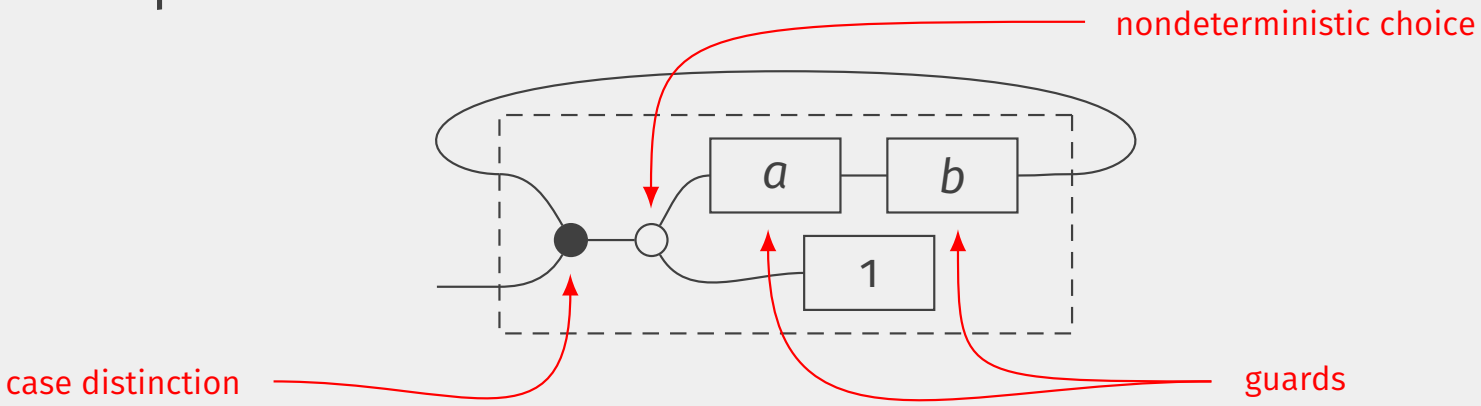
$$\begin{aligned} a(ba)^\star b + 1 &= a((ba)(ba)^\star + 1)b + 1 \\ &= a(ba)(ba)^\star b + a1b + 1 \\ &= (ab)a(ba)^\star b + ab + 1 \\ &= (ab)(a(ba)^\star b + 1) + 1 \end{aligned}$$

This only works because the map $x \mapsto abx + 1$ is guarded

Contrastingly, the map $x \mapsto (a + 1)x + 1$ is unguarded and has infinitely many fixpoints

Guarded Traces

Automata can be organized into a traced symmetric category, e.g. $(ab)^\star$ is expressed as

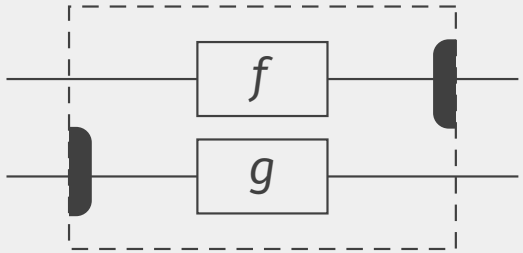


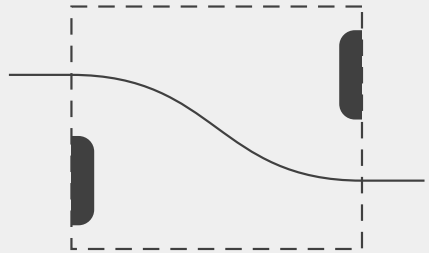
But we do not have an explicit distinction between guarded and unguarded loops

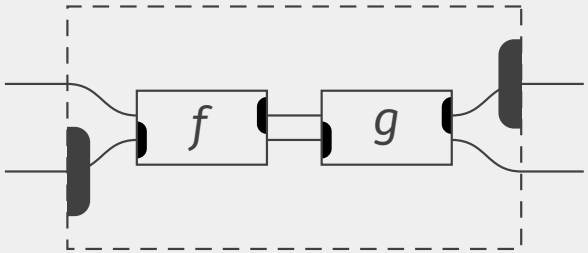
- Guarded traces are simpler and better behaved
- They often have special useful properties, e.g. uniqueness
- Sometimes, there is no candidate for a total trace operator

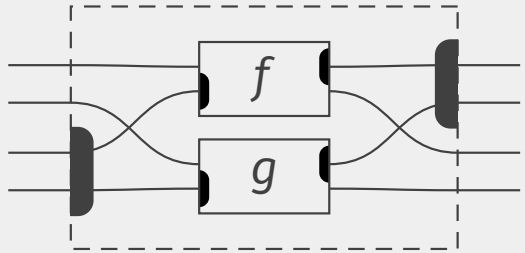
Guarded Categories

Guarded (monoidal) categories are defined in terms of decorated diagrams, subject to the axioms:





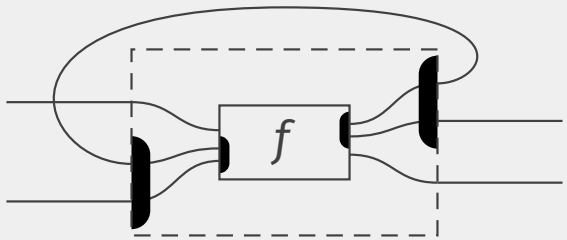




Only the paths from left bars to right bars are considered guarded

Guarded Traces

A guarded category is guarded traced if it supports guarded traces:



Intuitively, we are allowed to close the loop because it only runs through the interface of f indicated by the black bars

Total Guardedness and Vacuous Guardedness

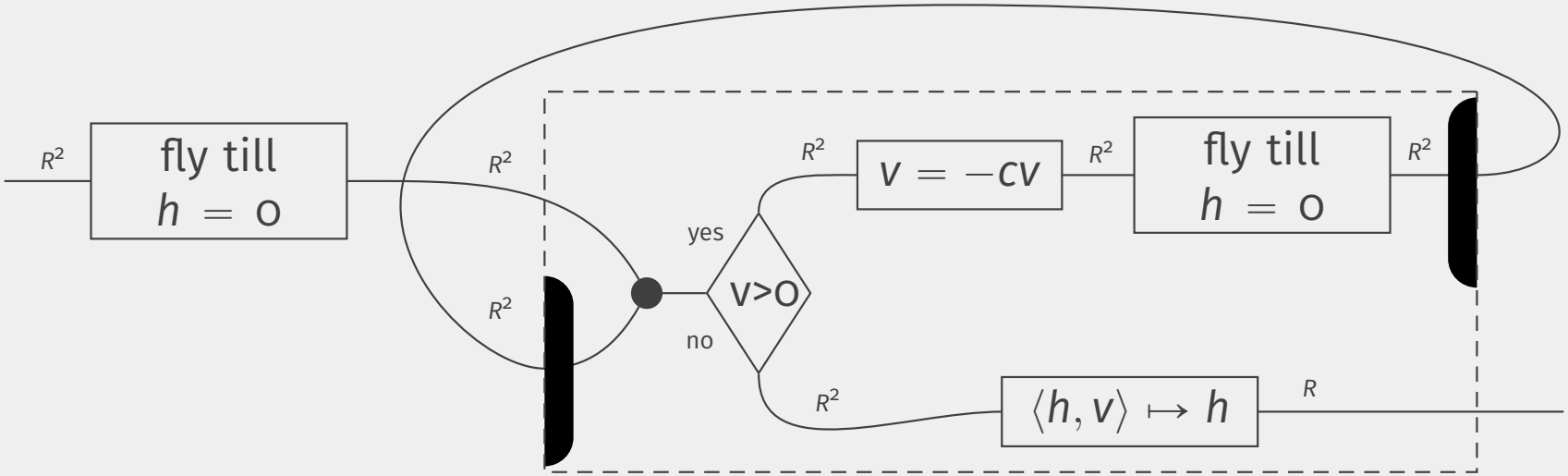
With guarded categories we mediate between symmetric monoidal categories and traced symmetric ones with

- vacuous guardedness: there are no guarded paths at all (symmetric monoidal categories)
- total guardedness: every path is guarded (traced symmetric categories)
- motivating examples (such as automata) occur properly between these two extremes

Bouncing Ball Revisited

Guardedness in hybrid semantics means progressiveness, i.e. consuming non-zero time

The behaviour of bouncing ball is described in terms of velocity and height $\langle v, h \rangle \in R^2$:



In contrast to general traces, guarded ones are computed as least fixpoints

Further Examples

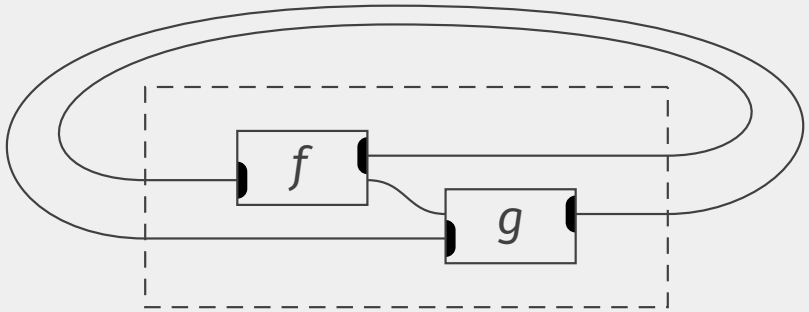
- Complete metric spaces: guardedness = contractiveness, fixpoints are computed via Banach's fixpoint theorem
- Infinite-dimentional Hilbert spaces under vacuous guardedness, traces = traces of bounded linear operators
- Infinite trace semantics: guardedness by actions, traces are greatest fixpoints, e.g. $a^\star + a^\omega$ is the canonical fixpoint of

$$x \mapsto ax + 1,$$

and not just $a^\star$

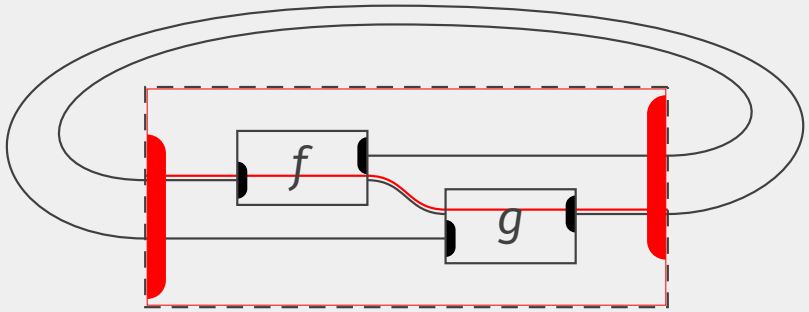
Completeness Problem

Q: Can we make sense of the following diagram?



Completeness Problem

Q: Can we make sense of the following diagram?

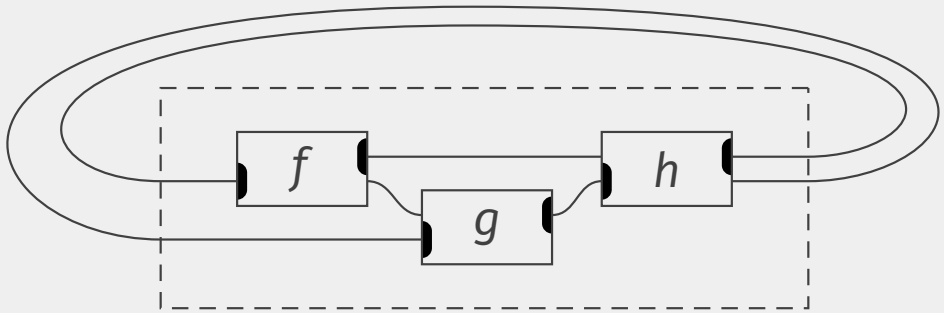


Completeness Problem

Q: Can we make sense of the following diagram?
A: Yes, because we can redraw it as

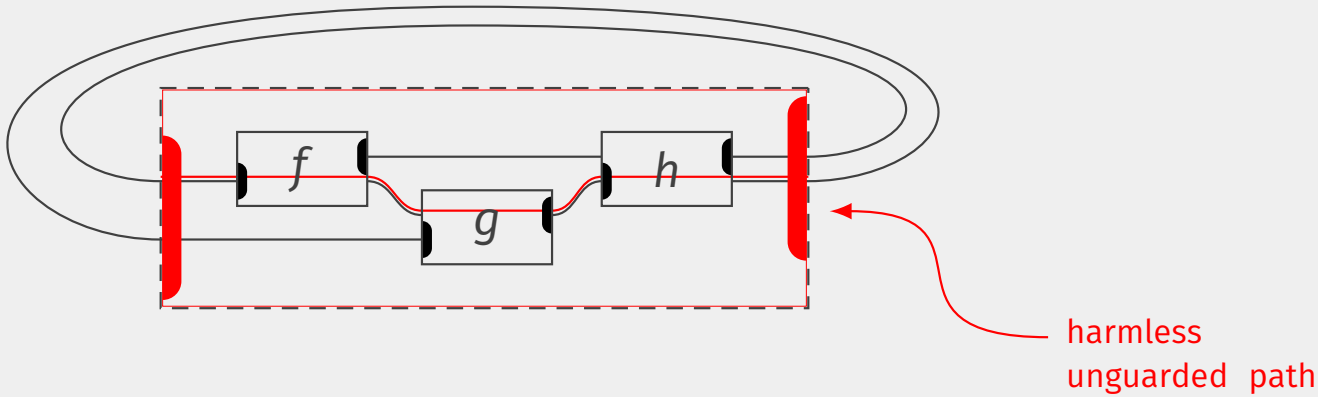
Completeness Problem

Q: What about this one:



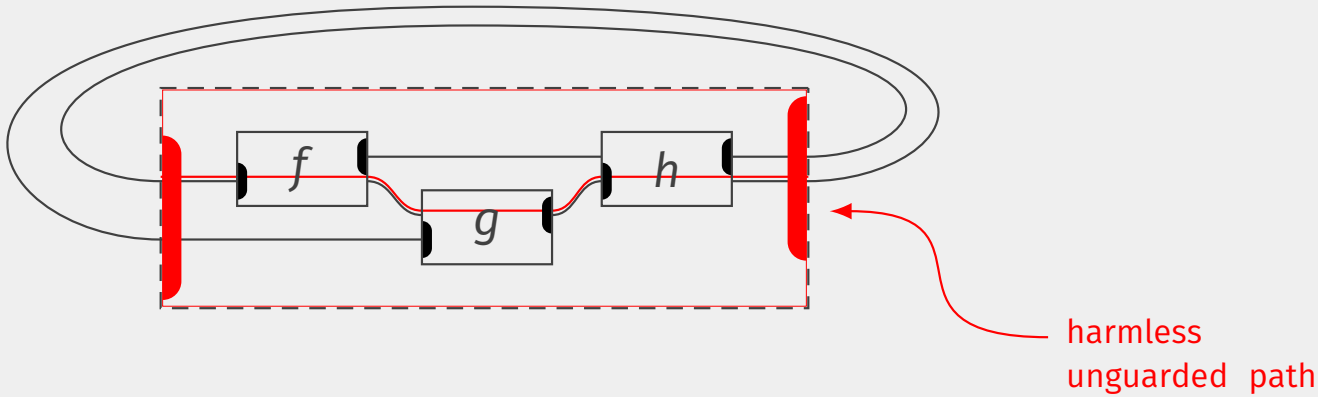
Completeness Problem

Q: What about this one:



Completeness Problem

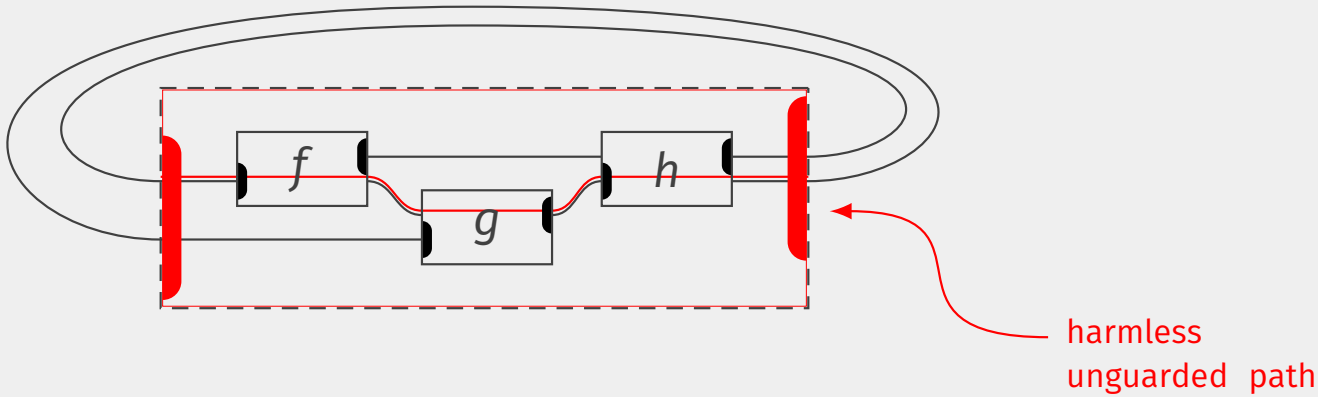
Q: What about this one:



Morally, this diagram should be OK, but we cannot rearrange it so as to be able to derive it from the axioms of guardedness

Completeness Problem

Q: What about this one:



Morally, this diagram should be OK, but we cannot rearrange it so as to be able to derive it from the axioms of guardedness

Open Problem: resolve the discrepancy between geometric guardedness and structural guardedness

Conclusions

- Guarded traced categories are abound in semantics and beyond
- Abstract notion of guardedness unifies various principles behind partiality of feedback: delay, progress, contractivity, etc
- Guarded categories come together with a semantically justified unifying diagrammatic metalanguage, suitable for visual programming and modelling (e.g. hybrid systems)
- Abstract guardedness helps to identify well-behaved notions of feedback (unique, least, greatest), which come together with the corresponding reasoning principles (co-induction, Scott induction)