Master's Thesis in Computational Logic

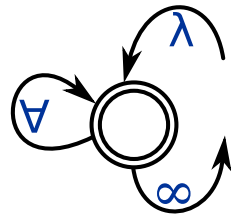
Infinitary Rewriting

Theory and Applications

Patrick Bahr

FAKULTÄT FÜR !INFORMATIK

Technische Universität Wien
Institut für Computersprachen
Arbeitsbereich: Theoretische Informatik und Logik
Betreuer: Ao. Univ. Prof. Dr. Bernhard Gramlich



September 23, 2009

Outline

Outline

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0)$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$s^2(0) * s^2(0)$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to s^2(0) + (s^2(0) * s(0))$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to s^2(0) + (s^2(0) * s(0))$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to^2 s^2(0) + (s^2(0) + (s^2(0) * 0))$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to^2 s^2(0) + (s^2(0) + (s^2(0) * 0))$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to^3 s^2(0) + (s^2(0) + 0)$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to^3 s^2(0) + (s^2(0) + 0)$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to^4 s^2(0) + s^2(0)$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to^4 s^2(0) + s^2(0)$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to^5 s(s^2(0) + s(0))$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to^5 s(s^2(0) + s(0))$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to^6 s(s(s^2(0) + 0))$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to^6 s(s(s^2(0) + 0))$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to^7 s(s(s^2(0)))$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to^7 s^4(0)$$

Rewriting Systems

What are rewriting systems?

- consist of directed symbolic equations over objects such as strings, terms, graphs etc.
- based on the idea of replacing equals by equals
- provide a formal model of computation
- term rewriting is the foundation of functional programming

Example (Term rewriting system defining addition and multiplication)

$$\mathcal{R}_{+*} = \begin{cases} x + 0 \to x & x * 0 \to 0 \\ x + s(y) \to s(x + y) & x * s(y) \to x + (x * y) \end{cases}$$

$$s^2(0) * s^2(0) \to^7 s^4(0)$$

$\mathcal{R}_{+*}$ is terminating!

Non-Terminating Rewriting Systems

Termination guarantees that every reduction sequence leads to a normal form, i.e. a final outcome.

Non-Terminating Rewriting Systems

Termination guarantees that every reduction sequence leads to a normal form, i.e. a final outcome.

Non-terminating systems can be meaningful

- modelling reactive systems, e.g. by process calculi
- approximation algorithms which enhance the accuracy of the approximation with each iteration, e.g. computing π
- specification of infinite data structures, e.g. streams

Non-Terminating Rewriting Systems

Termination guarantees that every reduction sequence leads to a normal form, i.e. a final outcome.

Non-terminating systems can be meaningful

- modelling reactive systems, e.g. by process calculi
- approximation algorithms which enhance the accuracy of the approximation with each iteration, e.g. computing π
- specification of infinite data structures, e.g. streams

Example (Infinite lists)

$$\mathcal{R}_{nats} = \left\{ \quad from(x) \to x : from(s(x)) \right.$$

$$from(0)$$

Non-Terminating Rewriting Systems

Termination guarantees that every reduction sequence leads to a normal form, i.e. a final outcome.

Non-terminating systems can be meaningful

- modelling reactive systems, e.g. by process calculi
- approximation algorithms which enhance the accuracy of the approximation with each iteration, e.g. computing π
- specification of infinite data structures, e.g. streams

Example (Infinite lists)

$$\mathcal{R}_{nats} = \left\{ \quad from(x) \to x : from(s(x)) \right.$$

$$from(0) \to 0 : from(1)$$

Non-Terminating Rewriting Systems

Termination guarantees that every reduction sequence leads to a normal form, i.e. a final outcome.

Non-terminating systems can be meaningful

- modelling reactive systems, e.g. by process calculi
- approximation algorithms which enhance the accuracy of the approximation with each iteration, e.g. computing π
- specification of infinite data structures, e.g. streams

Example (Infinite lists)

$$\mathcal{R}_{nats} = \left\{ \quad from(x) \to x : from(s(x)) \right.$$

$$from(0) \;\to^2 0 : 1 : from(2)$$

Non-Terminating Rewriting Systems

Termination guarantees that every reduction sequence leads to a normal form, i.e. a final outcome.

Non-terminating systems can be meaningful

- modelling reactive systems, e.g. by process calculi
- approximation algorithms which enhance the accuracy of the approximation with each iteration, e.g. computing π
- specification of infinite data structures, e.g. streams

Example (Infinite lists)

$$\mathcal{R}_{nats} = \left\{ \quad from(x) \to x : from(s(x)) \right.$$

$$from(0) \to^3 0 : 1 : 2 : from(3)$$

Non-Terminating Rewriting Systems

Termination guarantees that every reduction sequence leads to a normal form, i.e. a final outcome.

Non-terminating systems can be meaningful

- modelling reactive systems, e.g. by process calculi
- approximation algorithms which enhance the accuracy of the approximation with each iteration, e.g. computing π
- specification of infinite data structures, e.g. streams

Example (Infinite lists)

$$\mathcal{R}_{nats} = \left\{ \quad from(x) \to x : from(s(x)) \right.$$

$$from(0) \to^4 0 : 1 : 2 : 3 : from(4)$$

Non-Terminating Rewriting Systems

Termination guarantees that every reduction sequence leads to a normal form, i.e. a final outcome.

Non-terminating systems can be meaningful

- modelling reactive systems, e.g. by process calculi
- approximation algorithms which enhance the accuracy of the approximation with each iteration, e.g. computing π
- specification of infinite data structures, e.g. streams

Example (Infinite lists)

$$\mathcal{R}_{nats} = \left\{ \quad from(x) \to x : from(s(x)) \right.$$

$$from(0) \;\to^5 0 : 1 : 2 : 3 : 4 : from(5)$$

Non-Terminating Rewriting Systems

Termination guarantees that every reduction sequence leads to a normal form, i.e. a final outcome.

Non-terminating systems can be meaningful

- modelling reactive systems, e.g. by process calculi
- approximation algorithms which enhance the accuracy of the approximation with each iteration, e.g. computing π
- specification of infinite data structures, e.g. streams

Example (Infinite lists)

$$\mathcal{R}_{nats} = \left\{ \quad from(x) \to x : from(s(x)) \right.$$

$$from(0) \;\to^6 0 : 1 : 2 : 3 : 4 : 5 : from(6)$$

Non-Terminating Rewriting Systems

Termination guarantees that every reduction sequence leads to a normal form, i.e. a final outcome.

Non-terminating systems can be meaningful

- modelling reactive systems, e.g. by process calculi
- approximation algorithms which enhance the accuracy of the approximation with each iteration, e.g. computing π
- specification of infinite data structures, e.g. streams

Example (Infinite lists)

$$\mathcal{R}_{nats} = \left\{ \quad from(x) \to x : from(s(x)) \right.$$

$$from(0) \to^6 0 : 1 : 2 : 3 : 4 : 5 : from(6) \to \ \ldots$$

Non-Terminating Rewriting Systems

Termination guarantees that every reduction sequence leads to a normal form, i.e. a final outcome.

Non-terminating systems can be meaningful

- modelling reactive systems, e.g. by process calculi
- approximation algorithms which enhance the accuracy of the approximation with each iteration, e.g. computing π
- specification of infinite data structures, e.g. streams

Example (Infinite lists)

$$\mathcal{R}_{nats} = \left\{ \quad from(x) \to x : from(s(x)) \right.$$

$$from(0) \;\to^6 0 : 1 : 2 : 3 : 4 : 5 : from(6) \to \; \ldots$$

intuitively this converges to the infinite list $0 : 1 : 2 : 3 : 4 : 5 : \ldots$.

Infinitary Rewriting

What is infinitary rewriting?

- formalises the outcome of an infinite reduction sequence
- allows reduction sequences of any ordinal number length
- deals with objects of possibly infinite size, i.e. infinite strings, terms, graphs etc.

Infinitary Rewriting

What is infinitary rewriting?

- formalises the outcome of an infinite reduction sequence
- allows reduction sequences of any ordinal number length
- deals with objects of possibly infinite size, i.e. infinite strings, terms, graphs etc.

Why consider infinitary rewriting?

- because we can

Infinitary Rewriting

What is infinitary rewriting?

- formalises the outcome of an infinite reduction sequence
- allows reduction sequences of any ordinal number length
- deals with objects of possibly infinite size, i.e. infinite strings, terms, graphs etc.

Why consider infinitary rewriting?

- because we can
- model for lazy functional programming
- semantics for non-terminating systems
- semantics for process algebras
- arises in cyclic term graph rewriting

Formalising Infinitary Term Rewriting

Complete metric on terms

Skip this

- terms are endowed with a complete metric in order to formalise the convergence of infinite reductions.
- metric distance between terms is inversely proportional to the shallowest depth at which they differ:

$$\mathbf{d}(s,t) = 2^{-\mathrm{sim}(s,t)}$$

$\mathrm{sim}(s,t)$ – depth of the shallowest discrepancy of s and t

Formalising Infinitary Term Rewriting

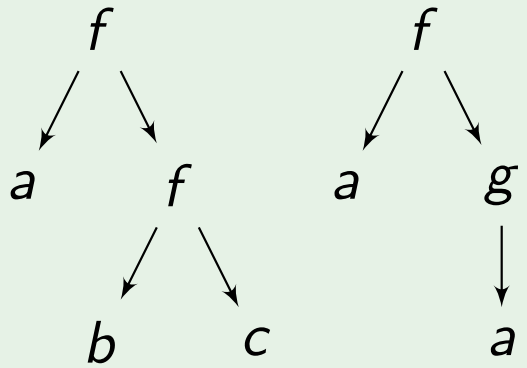
Complete metric on terms

- terms are endowed with a complete metric in order to formalise the convergence of infinite reductions.
- metric distance between terms is inversely proportional to the shallowest depth at which they differ:

$$\mathbf{d}(s,t) = 2^{-\mathrm{sim}(s,t)}$$

$\mathrm{sim}(s,t)$ – depth of the shallowest discrepancy of s and t

Example



Formalising Infinitary Term Rewriting

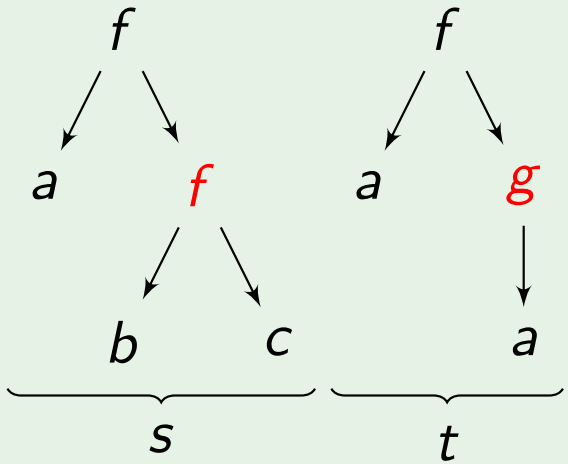
Complete metric on terms

- terms are endowed with a complete metric in order to formalise the convergence of infinite reductions.
- metric distance between terms is inversely proportional to the shallowest depth at which they differ:

$$\mathbf{d}(s,t) = 2^{-\mathrm{sim}(s,t)}$$

$\mathrm{sim}(s,t)$ – depth of the shallowest discrepancy of s and t

Example



Formalising Infinitary Term Rewriting

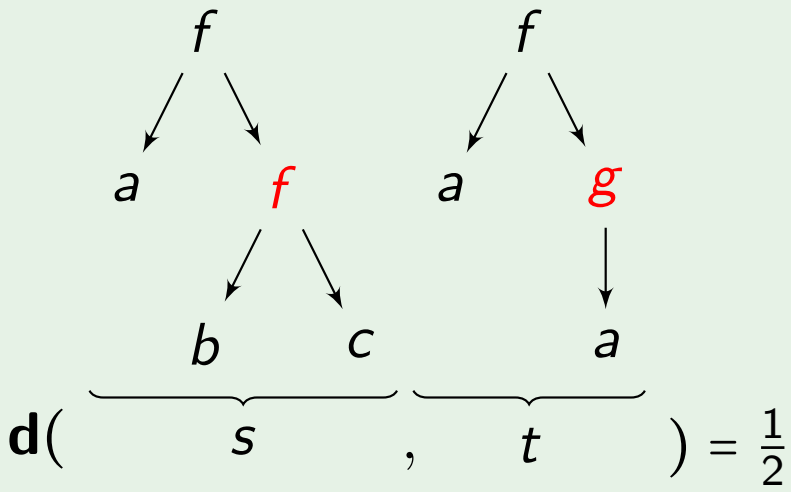
Complete metric on terms

- terms are endowed with a complete metric in order to formalise the convergence of infinite reductions.
- metric distance between terms is inversely proportional to the shallowest depth at which they differ:

$$\mathbf{d}(s,t) = 2^{-\mathrm{sim}(s,t)}$$

$\mathrm{sim}(s,t)$ – depth of the shallowest discrepancy of s and t

Example



Formalising Infinitary Term Rewriting

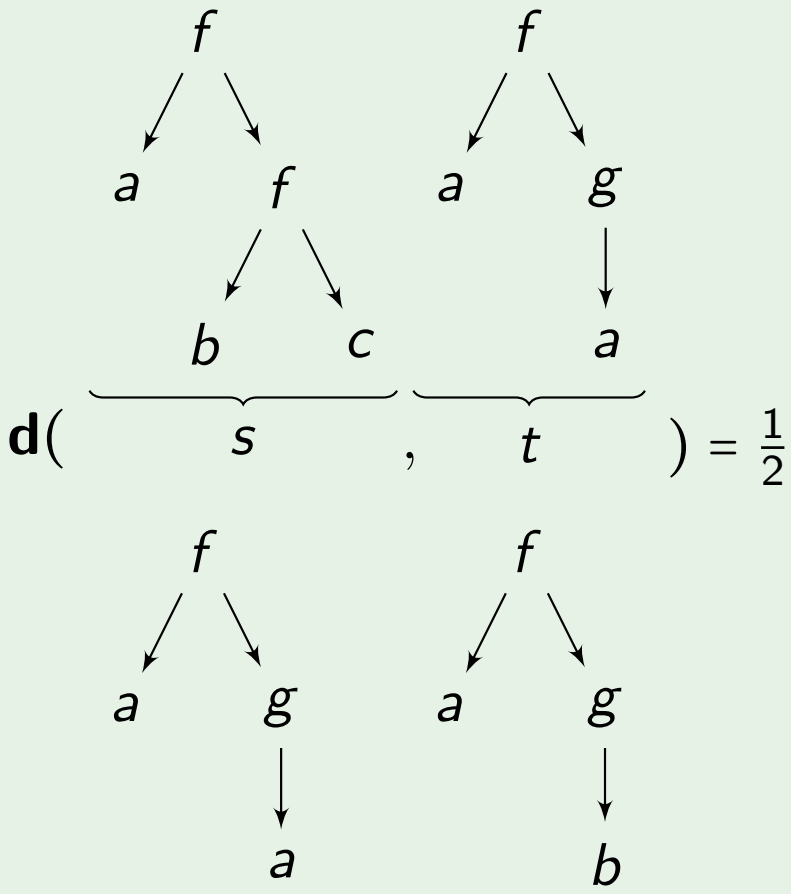
Complete metric on terms

- terms are endowed with a complete metric in order to formalise the convergence of infinite reductions.
- metric distance between terms is inversely proportional to the shallowest depth at which they differ:

$$\mathbf{d}(s,t) = 2^{-\mathrm{sim}(s,t)}$$

$\mathrm{sim}(s,t)$ – depth of the shallowest discrepancy of s and t

Example



Formalising Infinitary Term Rewriting

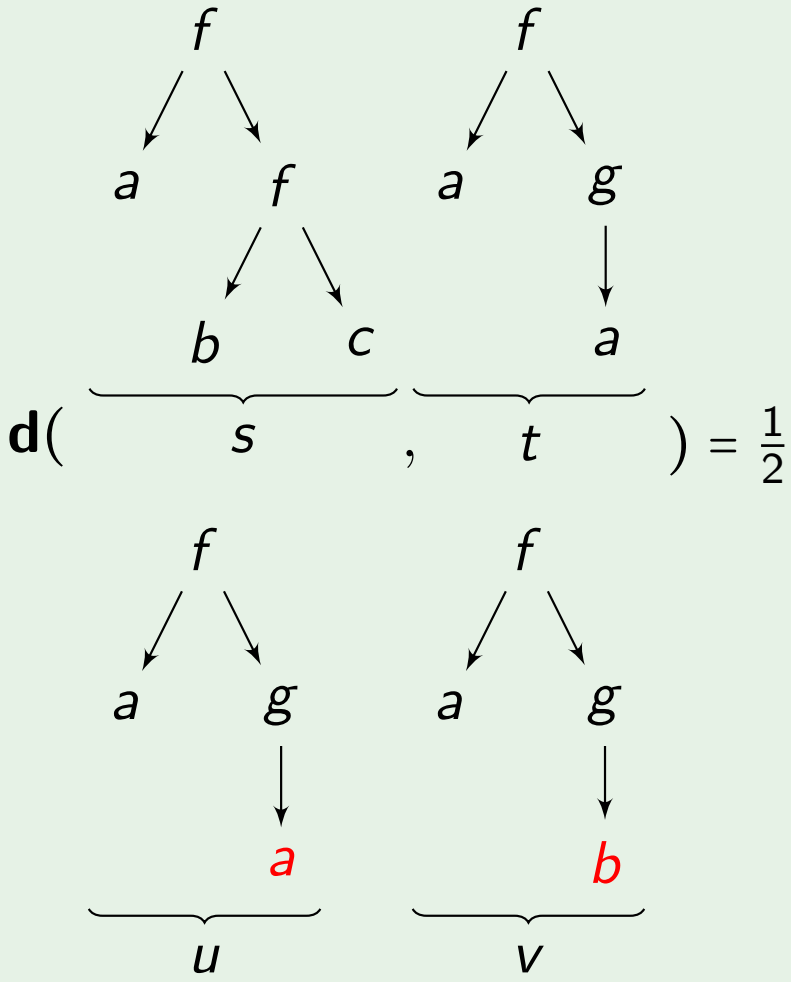
Complete metric on terms

- terms are endowed with a complete metric in order to formalise the convergence of infinite reductions.
- metric distance between terms is inversely proportional to the shallowest depth at which they differ:

$$\mathbf{d}(s,t) = 2^{-\mathrm{sim}(s,t)}$$

$\mathrm{sim}(s,t)$ – depth of the shallowest discrepancy of s and t

Example



Formalising Infinitary Term Rewriting

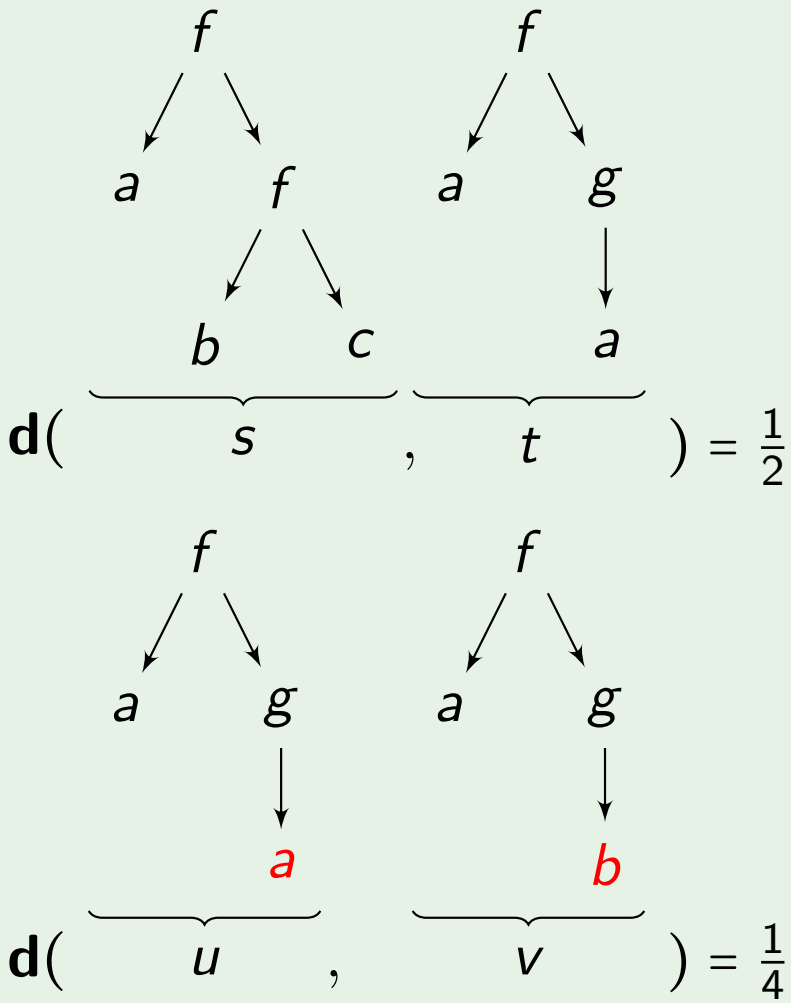
Complete metric on terms

- terms are endowed with a complete metric in order to formalise the convergence of infinite reductions.
- metric distance between terms is inversely proportional to the shallowest depth at which they differ:

$$\mathbf{d}(s,t) = 2^{-\mathrm{sim}(s,t)}$$

$\mathrm{sim}(s,t)$ – depth of the shallowest discrepancy of s and t

Example

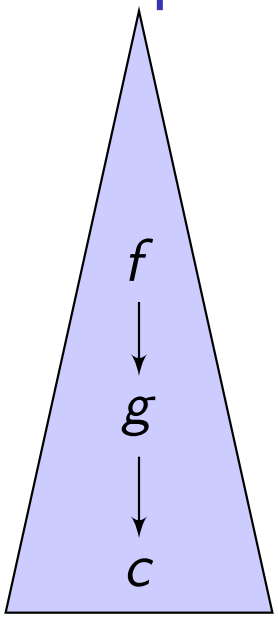


Convergence of Transfinite Reductions

Two different kinds of convergence

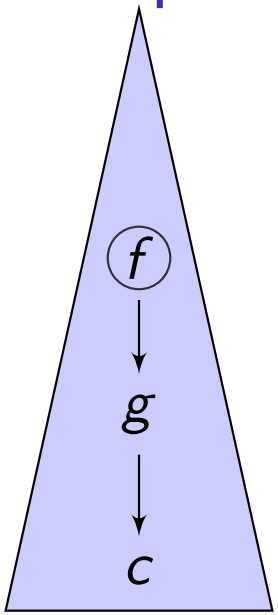
- weak convergence: convergence in the metric space of terms
  - $\rightsquigarrow$ for weak convergence the depth of the discrepancies of the terms has to tend to infinity
- strong convergence: convergence in the metric space + rewrite rules have to (eventually) be applied at increasingly large depth
  - $\rightsquigarrow$ for strong convergence the depth of where the rewrite rules are applied has to tend to infinity

Example: Weak Convergence



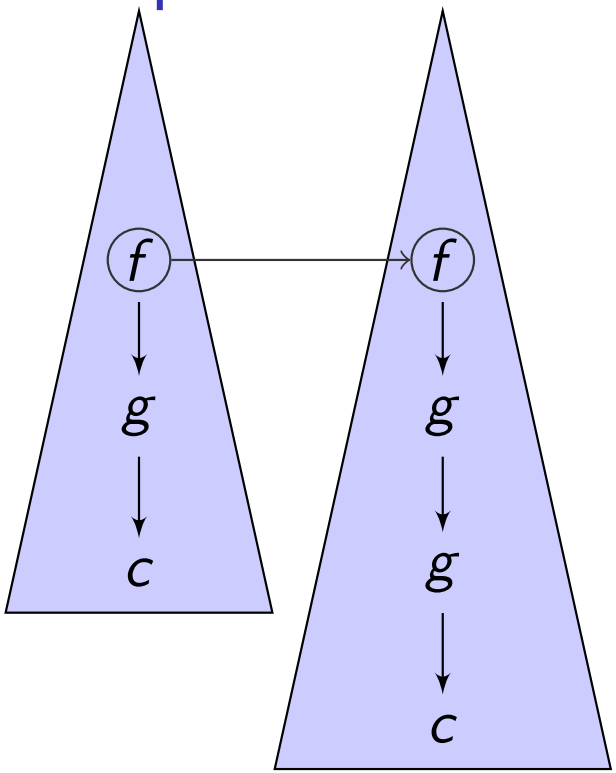
$$f(g(x)) \to f(g(g(x)))$$

Example: Weak Convergence



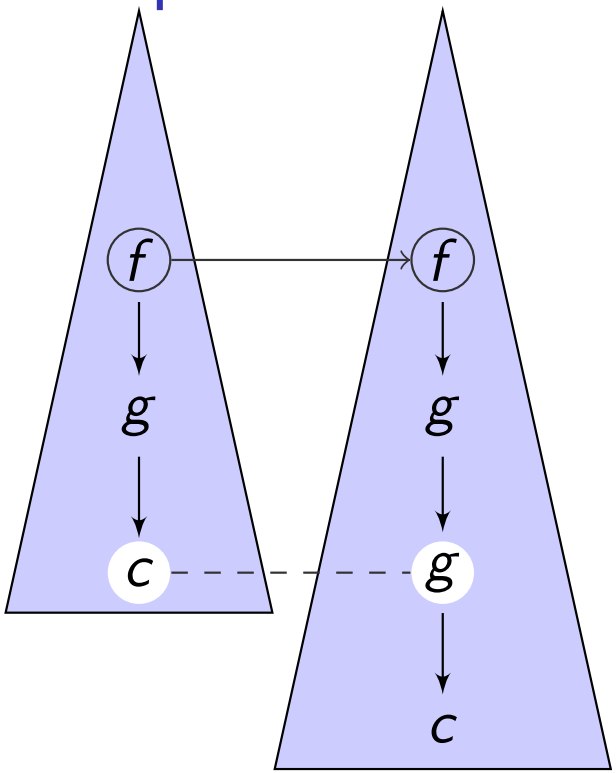
$$f(g(x)) \to f(g(g(x)))$$

Example: Weak Convergence



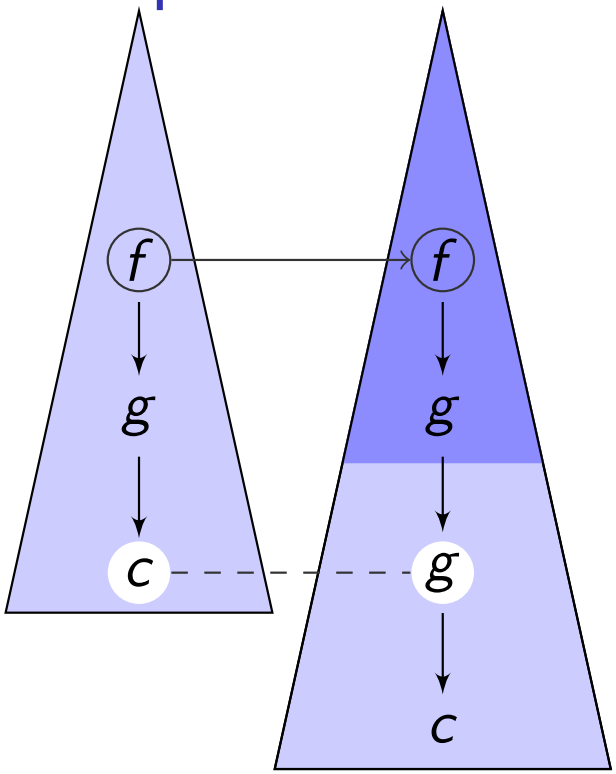
$$f(g(x)) \to f(g(g(x)))$$

Example: Weak Convergence



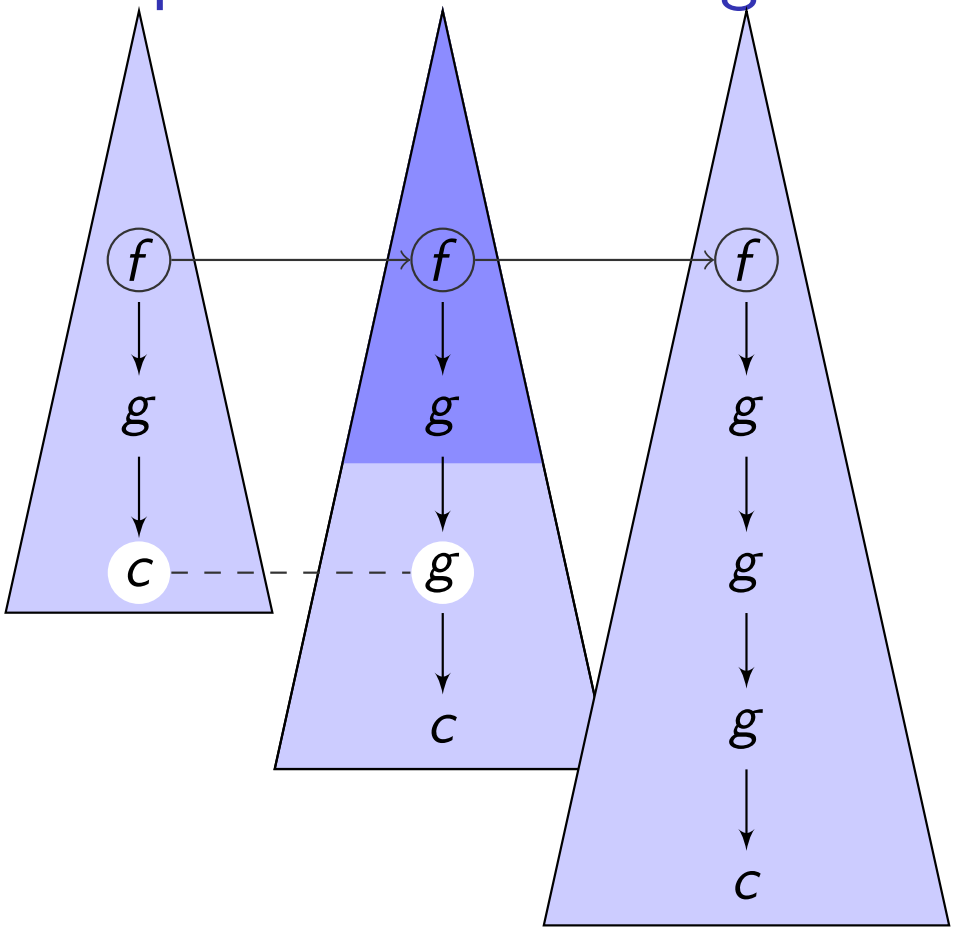
$$f(g(x)) \to f(g(g(x)))$$

Example: Weak Convergence



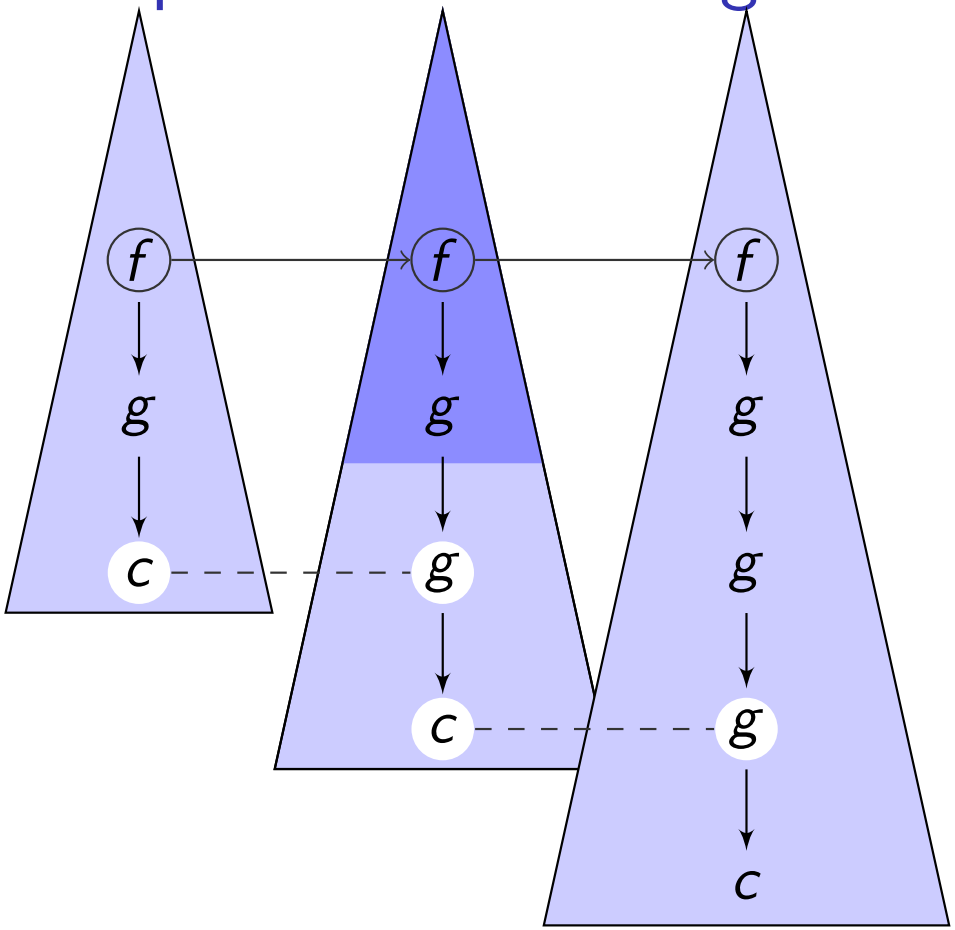
$$f(g(x)) \to f(g(g(x)))$$

Example: Weak Convergence



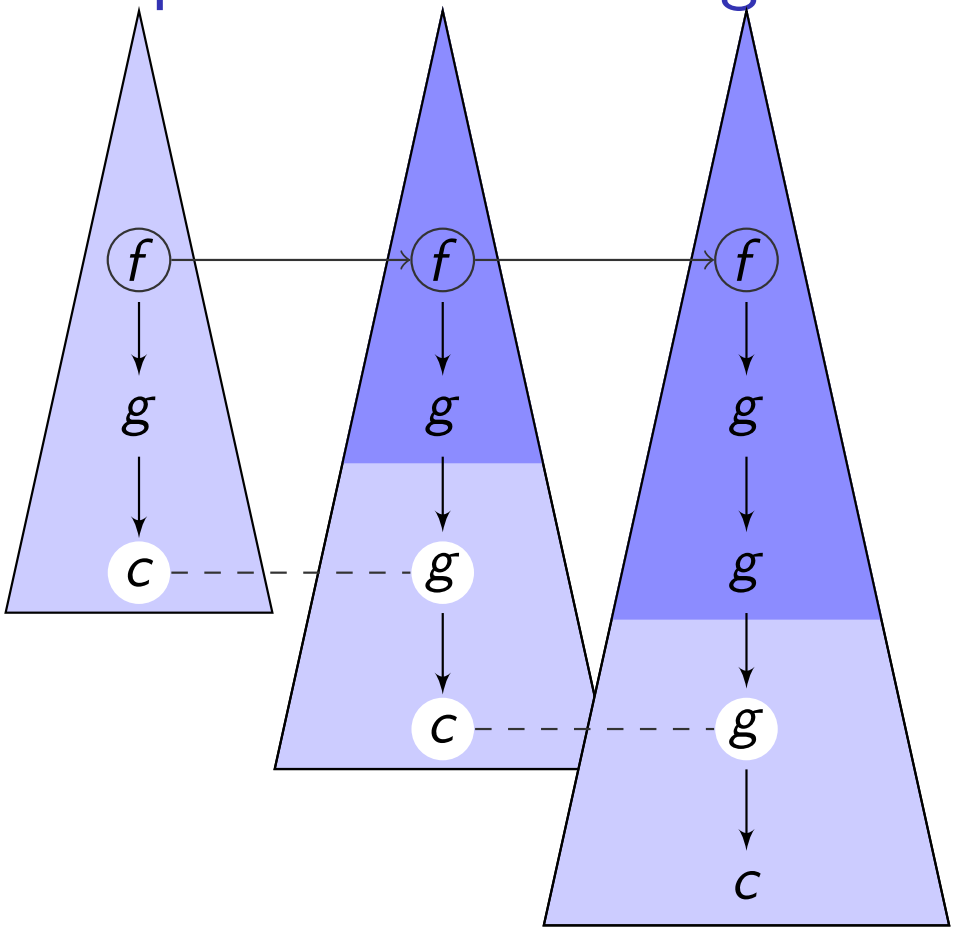
$$f(g(x)) \to f(g(g(x)))$$

Example: Weak Convergence



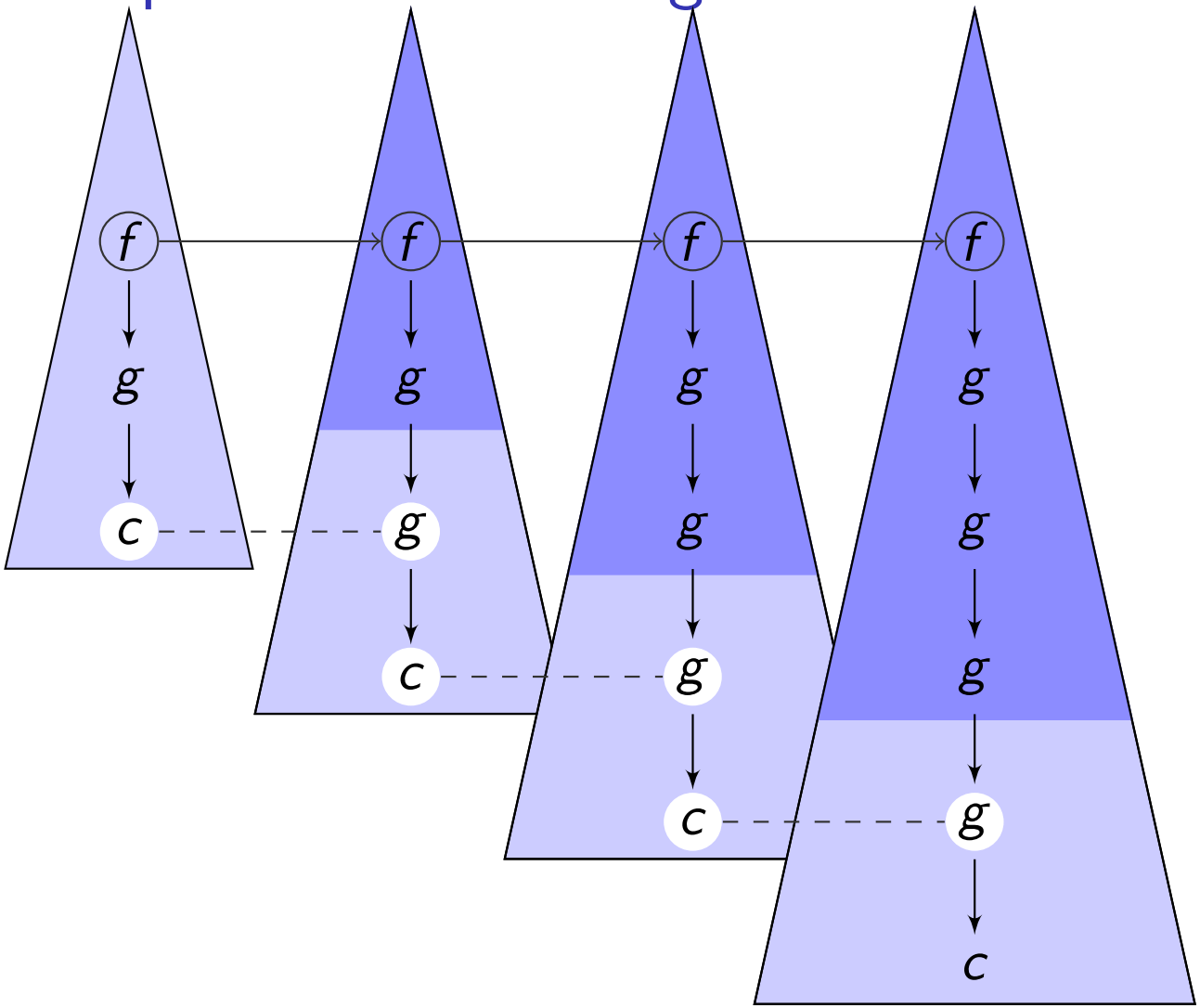
$$f(g(x)) \to f(g(g(x)))$$

Example: Weak Convergence



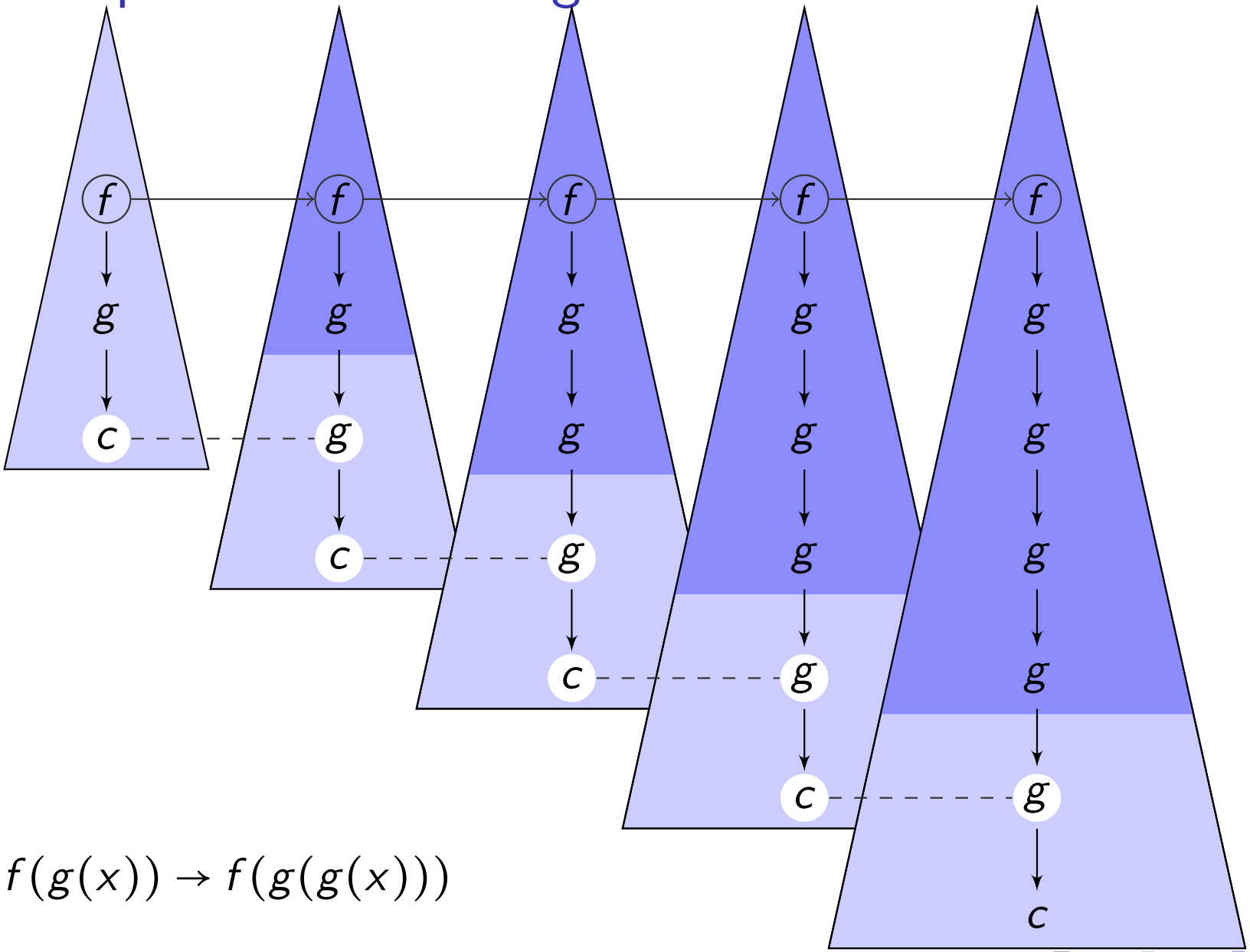
$$f(g(x)) \to f(g(g(x)))$$

Example: Weak Convergence



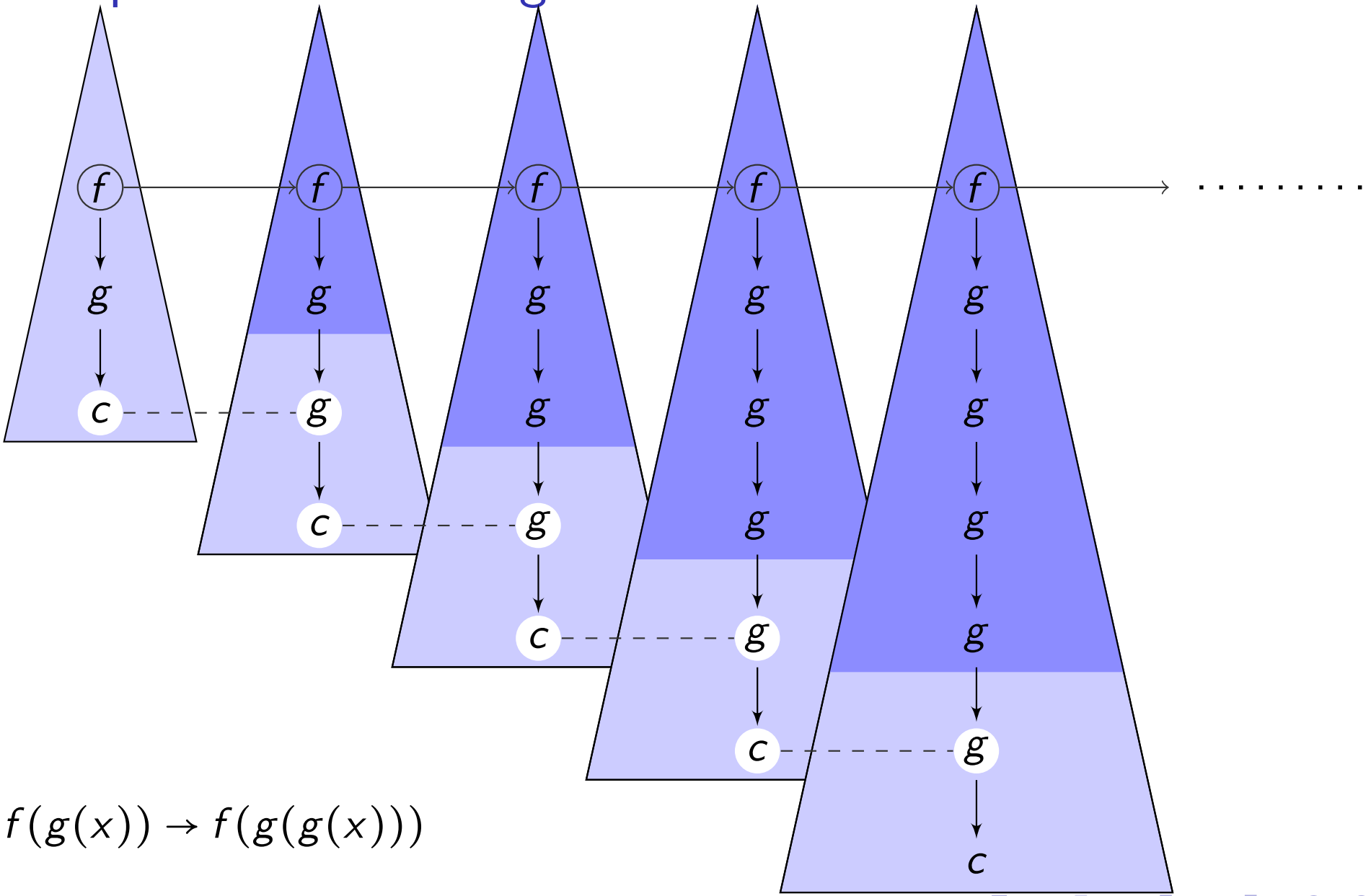
$$f(g(x)) \to f(g(g(x)))$$

Example: Weak Convergence



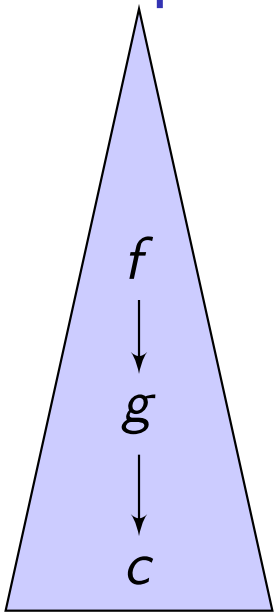
$$f(g(x)) \to f(g(g(x)))$$

Example: Weak Convergence



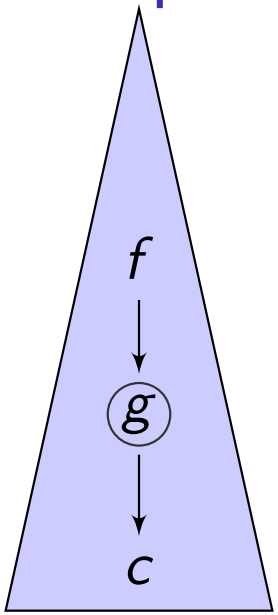
$$f(g(x)) \to f(g(g(x)))$$

Example: Strong Convergence



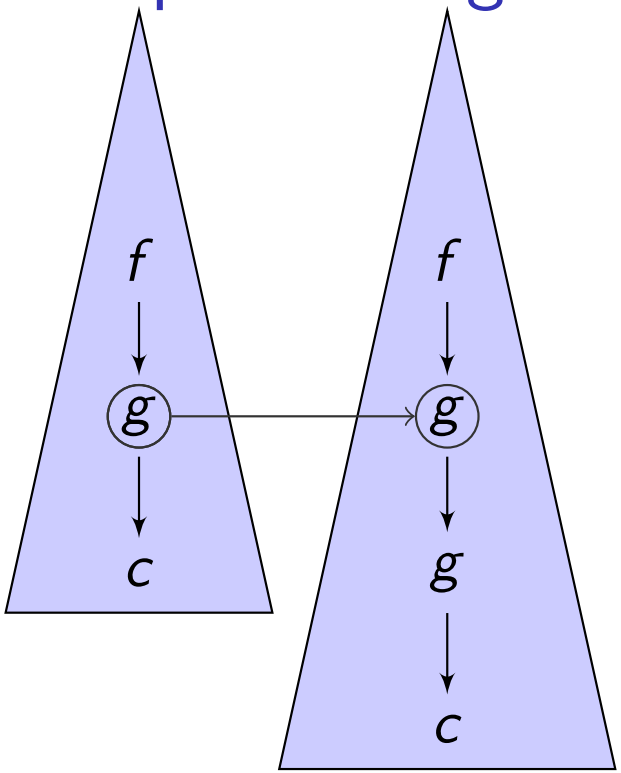
$$g(c) \to g(g(c))$$

Example: Strong Convergence



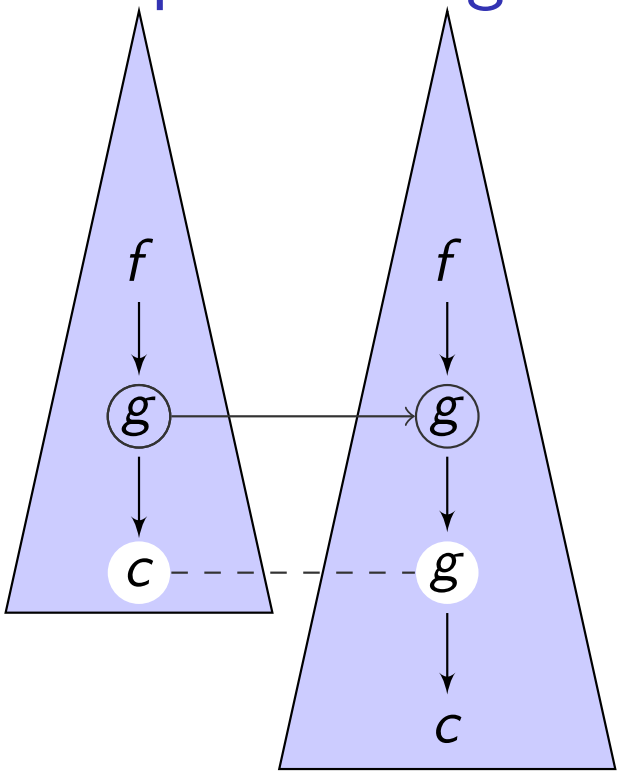
$$g(c) \to g(g(c))$$

Example: Strong Convergence



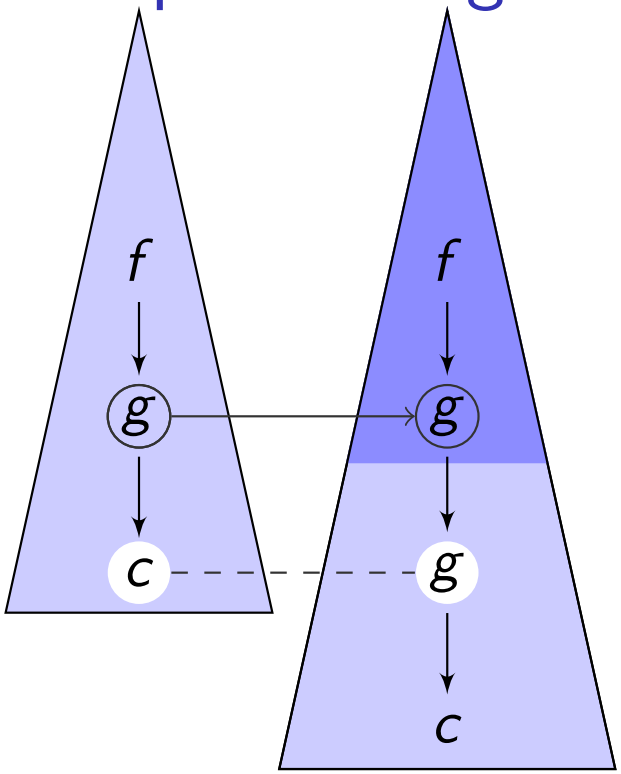
$$g(c) \to g(g(c))$$

Example: Strong Convergence



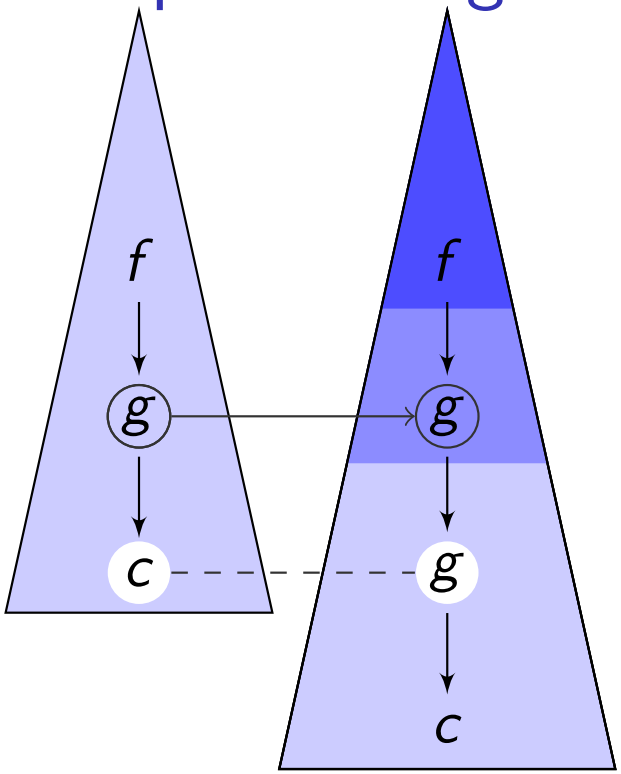
$$g(c) \to g(g(c))$$

Example: Strong Convergence



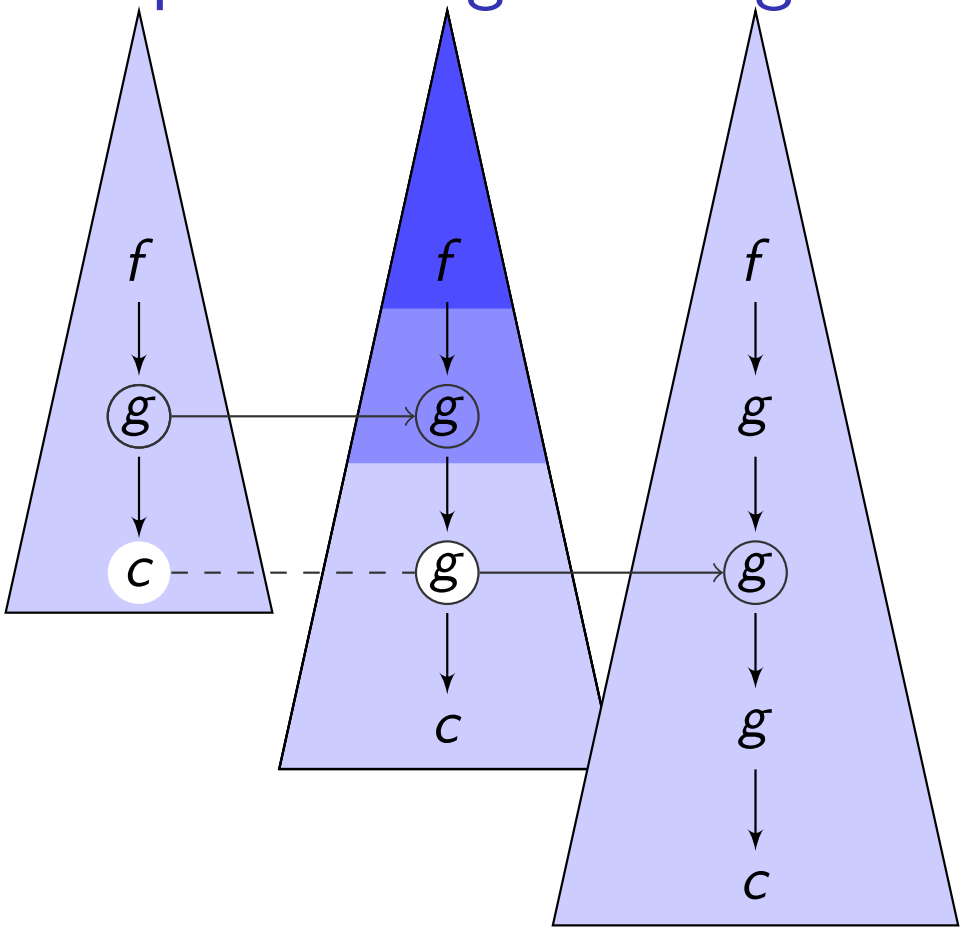
$$g(c) \to g(g(c))$$

Example: Strong Convergence



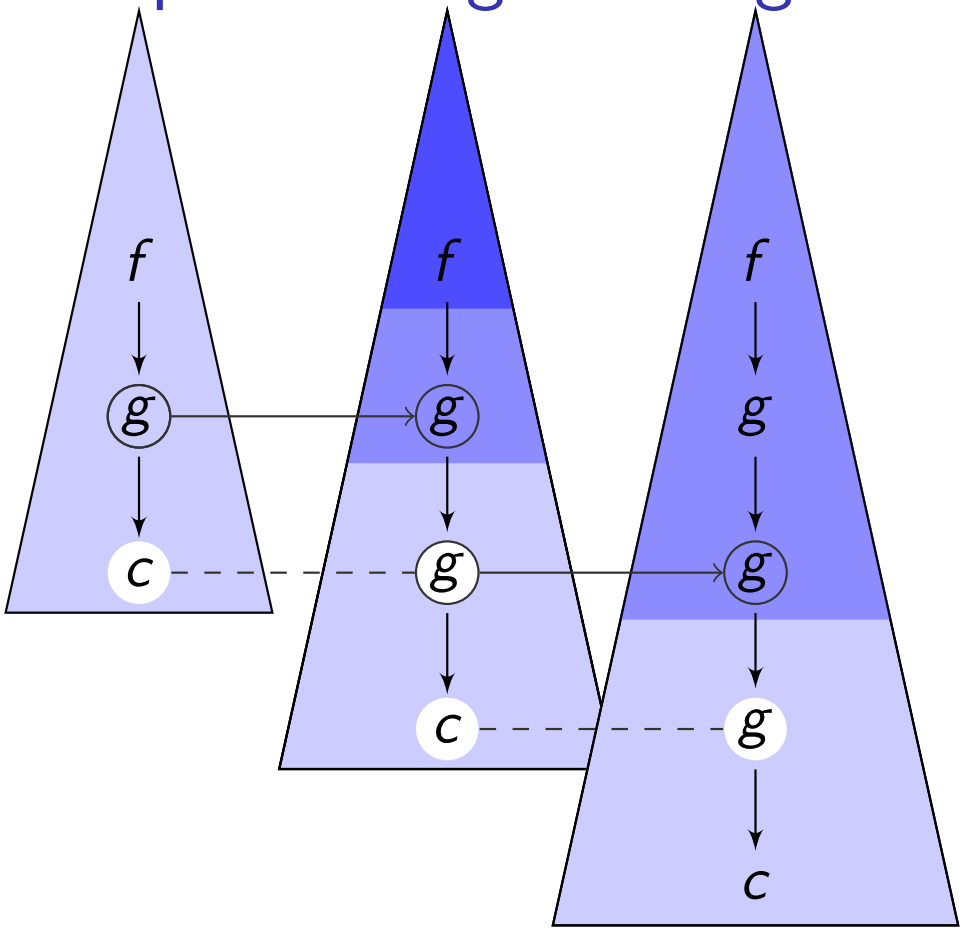
$$g(c) \to g(g(c))$$

Example: Strong Convergence



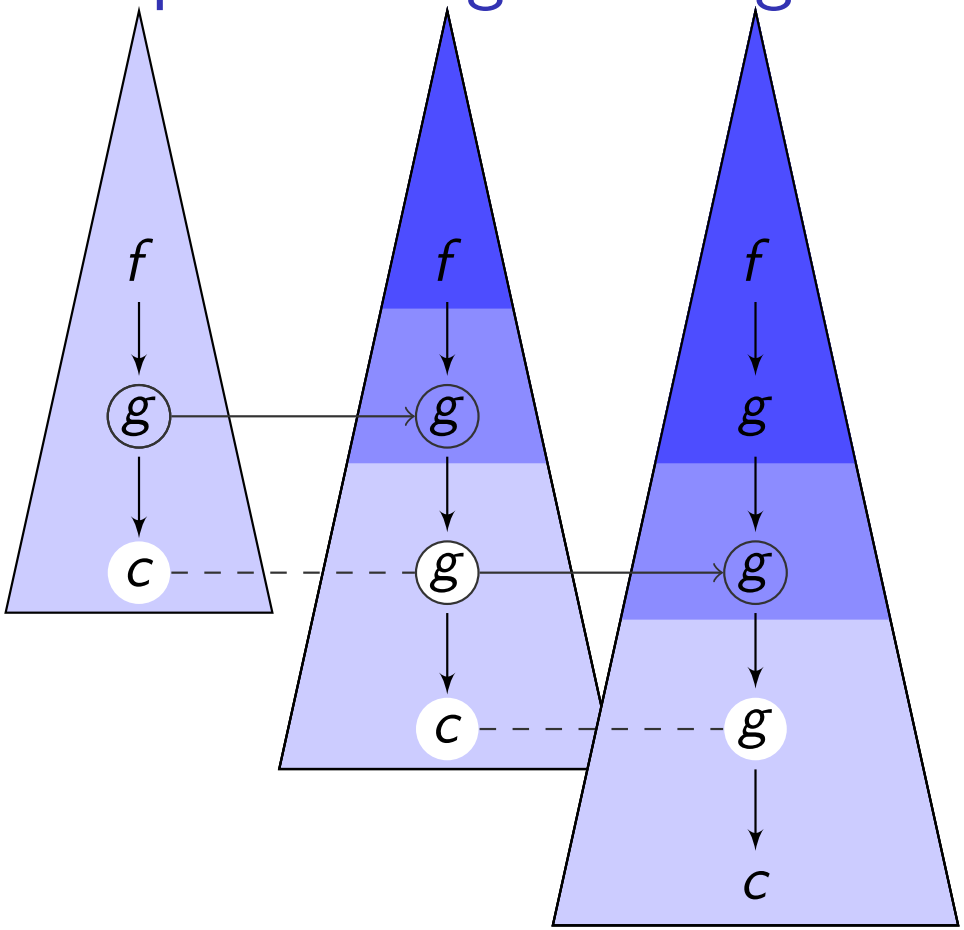
$g(c) \to g(g(c))$

Example: Strong Convergence



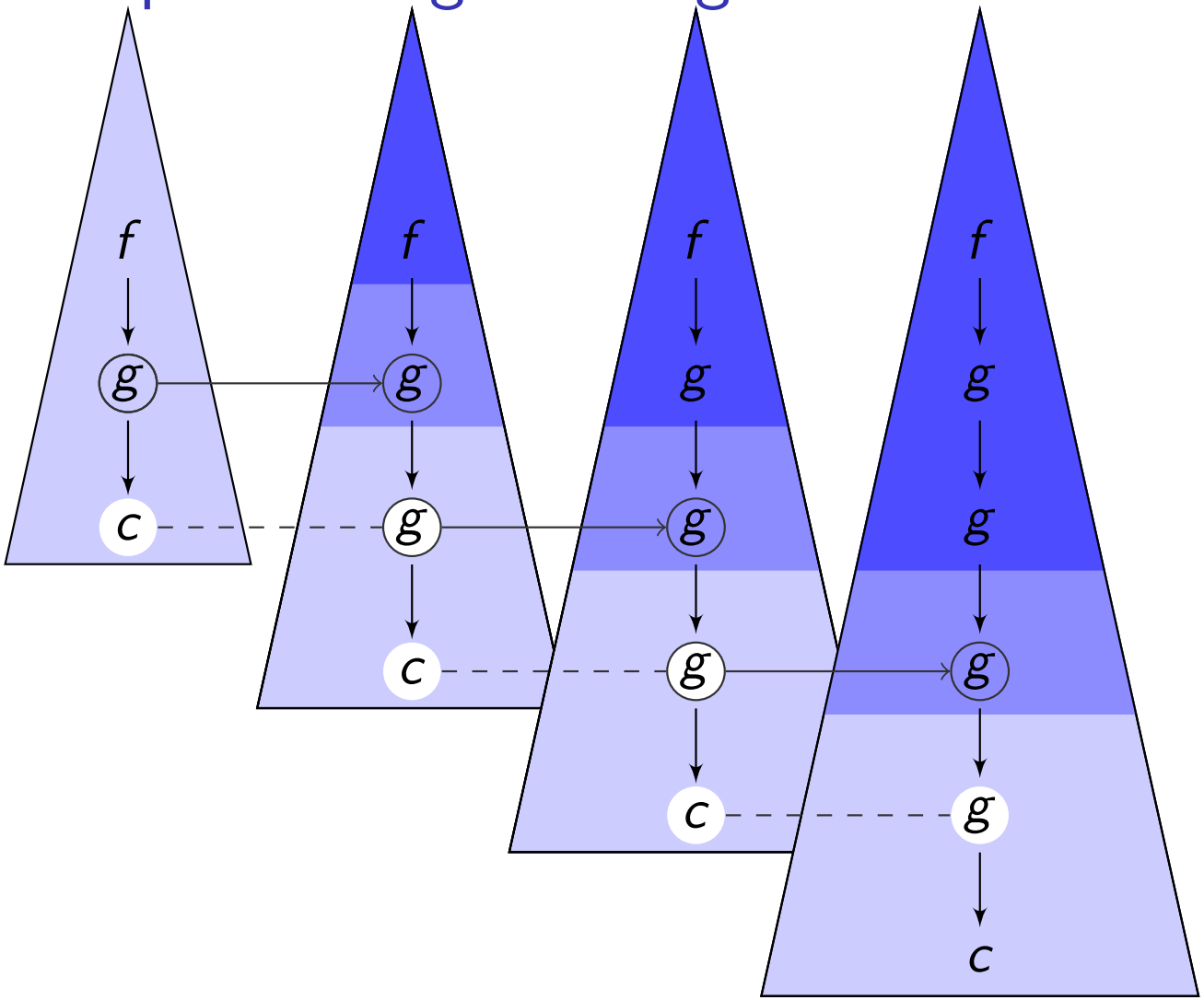
$$g(c) \to g(g(c))$$

Example: Strong Convergence



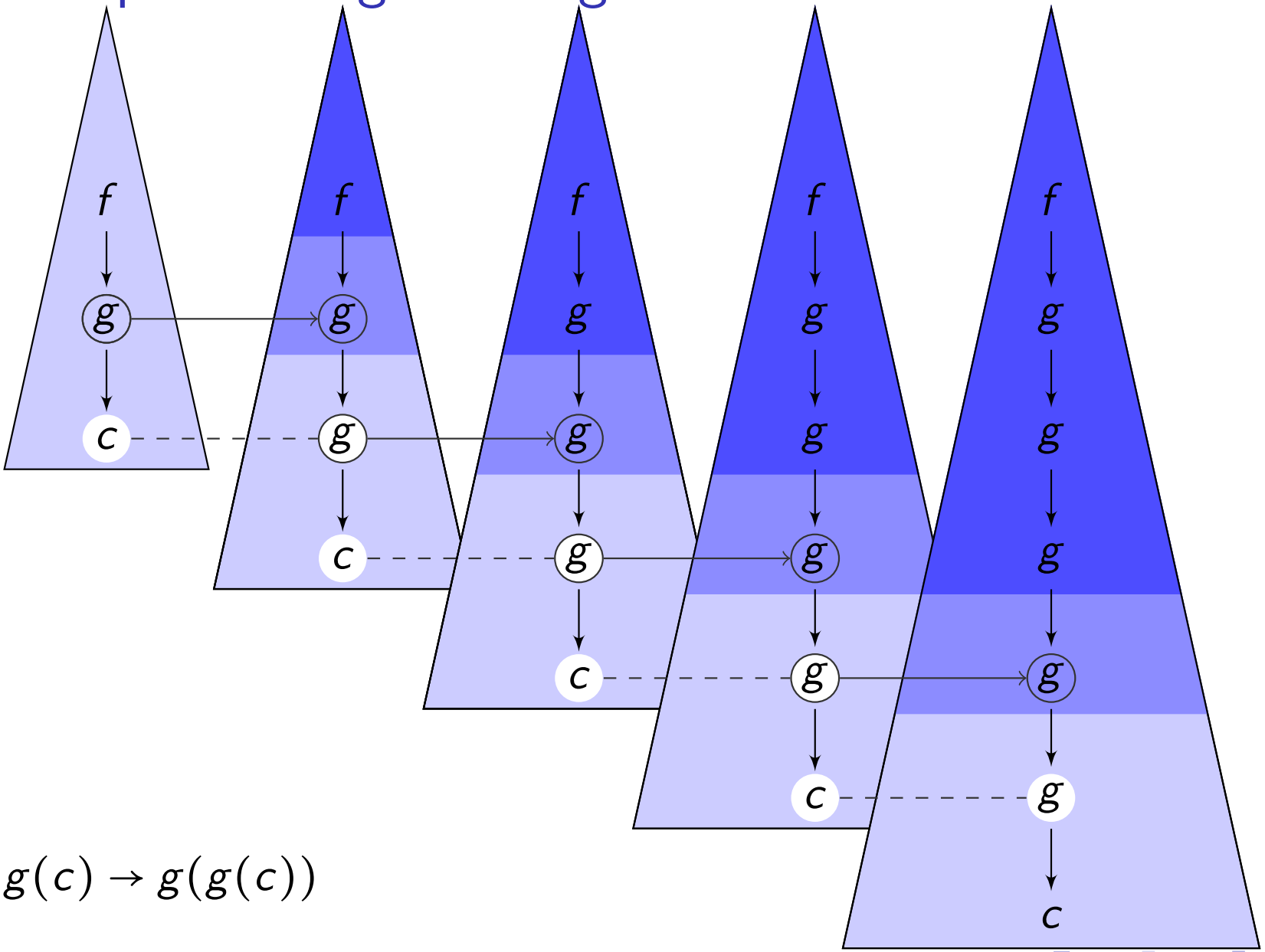
$g(c) \to g(g(c))$

Example: Strong Convergence



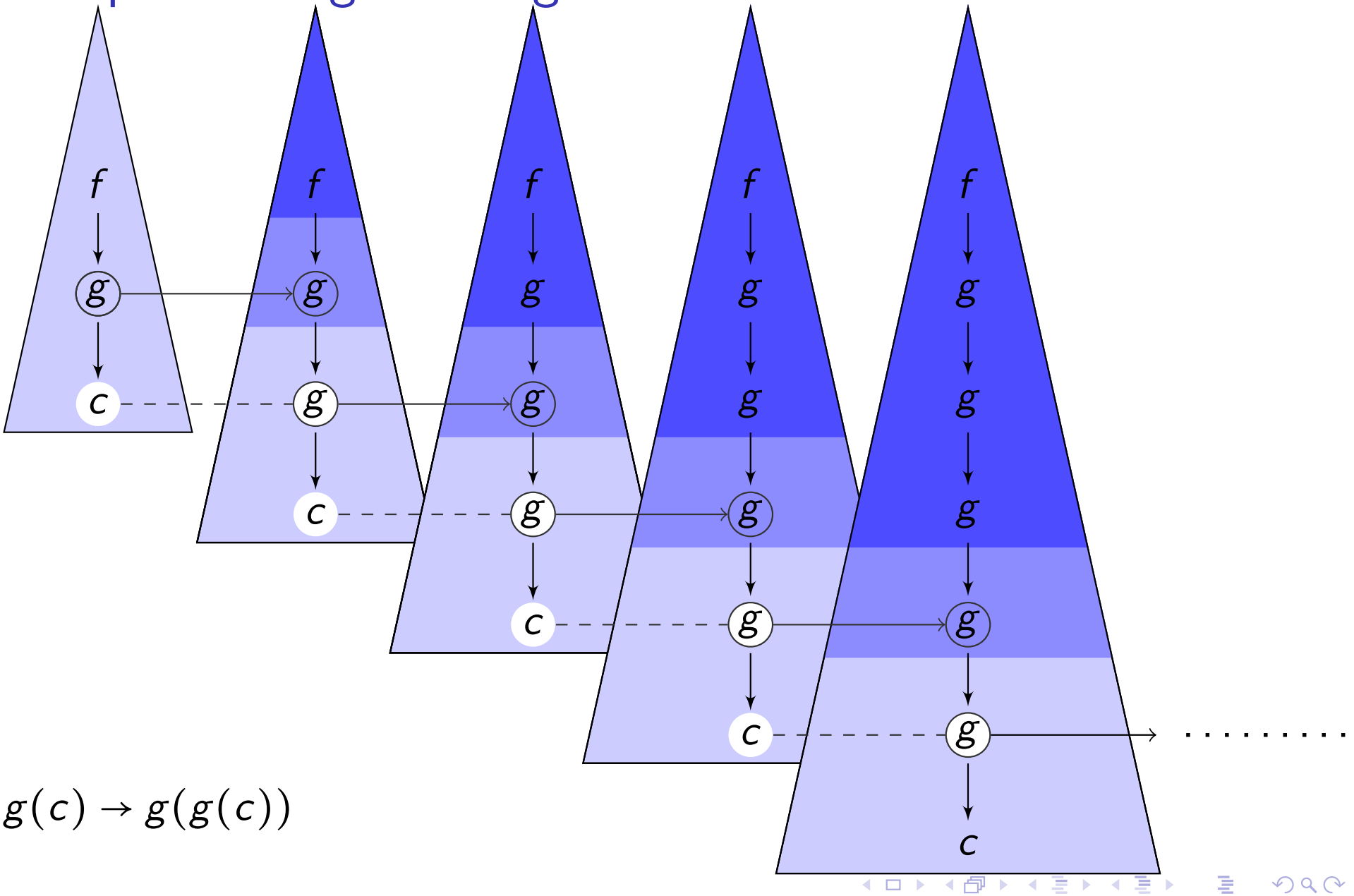
$g(c) \to g(g(c))$

Example: Strong Convergence



$g(c) \to g(g(c))$

Example: Strong Convergence



$g(c) \to g(g(c))$

Goals of the Thesis

Survey of the field of infinitary term rewriting

- infinitary versions of abstract reduction systems
- infinitary versions of finitary properties (confluence, termination etc.)
- relation to term graph rewriting
- applications (to finitary rewriting or other areas)

Goals of the Thesis

Survey of the field of infinitary term rewriting

- infinitary versions of abstract reduction systems
- infinitary versions of finitary properties (confluence, termination etc.)
- relation to term graph rewriting
- applications (to finitary rewriting or other areas)

Research directions

- extending the theory
- how to implement infinitary term rewriting?
- finitely represent infinite terms and reductions
- overcome drawbacks and limitations of infinitary term rewriting

Outline

Main Contributions

Partial order model of infinitary rewriting

- instead of a metric, a partial order is used to formalise convergence
  - $\rightsquigarrow$ more fine-grained model of convergence
- both the metric and the partial order model are investigated and compared on an abstract level

Main Contributions

Partial order model of infinitary rewriting

- instead of a metric, a partial order is used to formalise convergence
  - $\rightsquigarrow$ more fine-grained model of convergence
- both the metric and the partial order model are investigated and compared on an abstract level

Infinitary term rewriting w.r.t. the partial order model

- has more advantageous properties (confluence, normalisation)
- subsumes the metric model
- equivalent to Böhm reductions

Main Contributions

Partial order model of infinitary rewriting

- instead of a metric, a partial order is used to formalise convergence
  - $\rightsquigarrow$ more fine-grained model of convergence
- both the metric and the partial order model are investigated and compared on an abstract level

Infinitary term rewriting w.r.t. the partial order model

- has more advantageous properties (confluence, normalisation)
- subsumes the metric model
- equivalent to Böhm reductions

Infinitary term graph rewriting

- a complete metric and a complete semilattice are devised
- infinitary term graph rewriting is introduced and analysed

Partial Order Model of Infinitary Rewriting

Described on the example of terms

Partial order on terms

- partial terms: terms with additional constant $\bot$ (read as "undefined")
- partial order $\leq_\bot$ reads as: "is less defined than"
- $\leq_\bot$ is a complete semilattice ($=$ cpo $+$ glbs of non-empty sets)

Partial Order Model of Infinitary Rewriting

Described on the example of terms

Partial order on terms

- partial terms: terms with additional constant $\bot$ (read as "undefined")
- partial order $\leq_\bot$ reads as: "is less defined than"
- $\leq_\bot$ is a complete semilattice (= cpo + glbs of non-empty sets)

Convergence

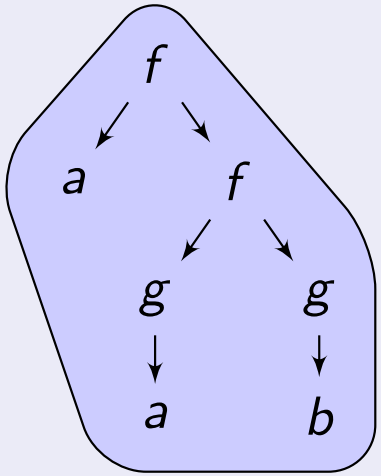
- formalised by the limit inferior:

$$\liminf_{\iota \to \alpha} t_\iota = \bigsqcup_{\beta < \alpha} \bigsqcap_{\beta \leq \iota < \alpha} t_\iota$$

- intuition: eventual persistence of nodes of the terms
- weak convergence: limit inferior of the terms of the reduction
- strong convergence: limit inferior of the contexts of the reduction

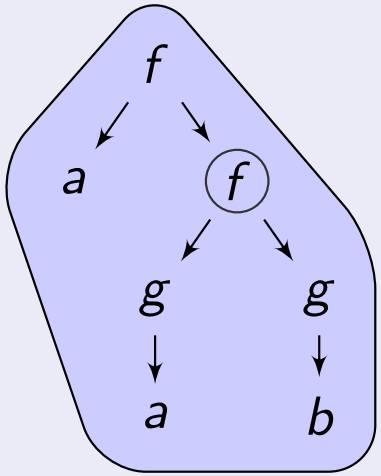
An Example

Reduction sequence for $f(x, y) \to f(y, x)$



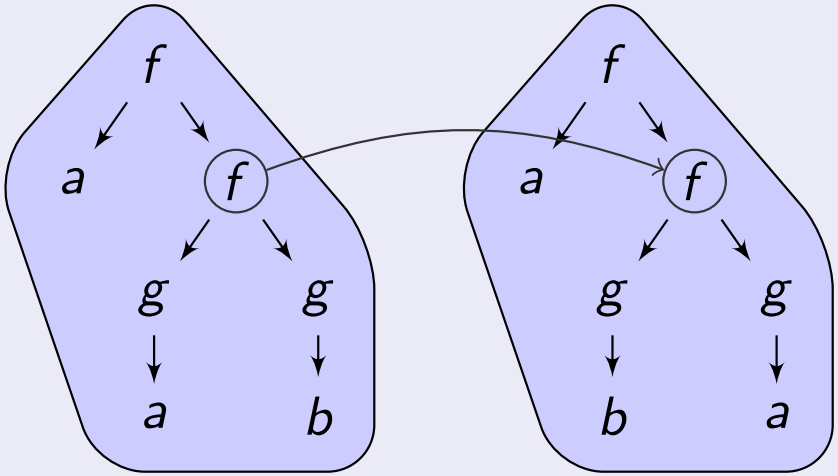
An Example

Reduction sequence for $f(x, y) \to f(y, x)$



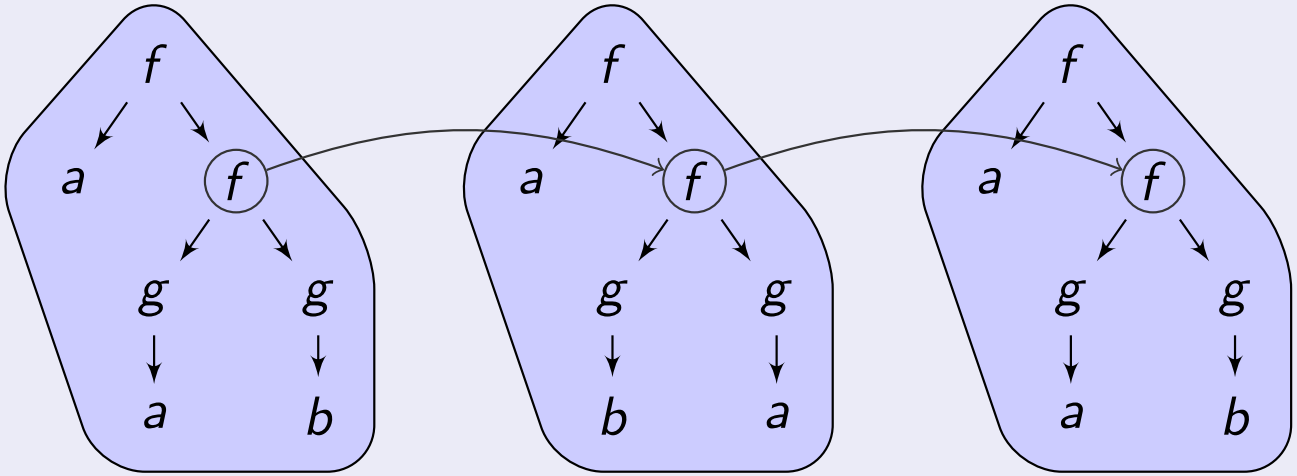
An Example

Reduction sequence for $f(x, y) \to f(y, x)$



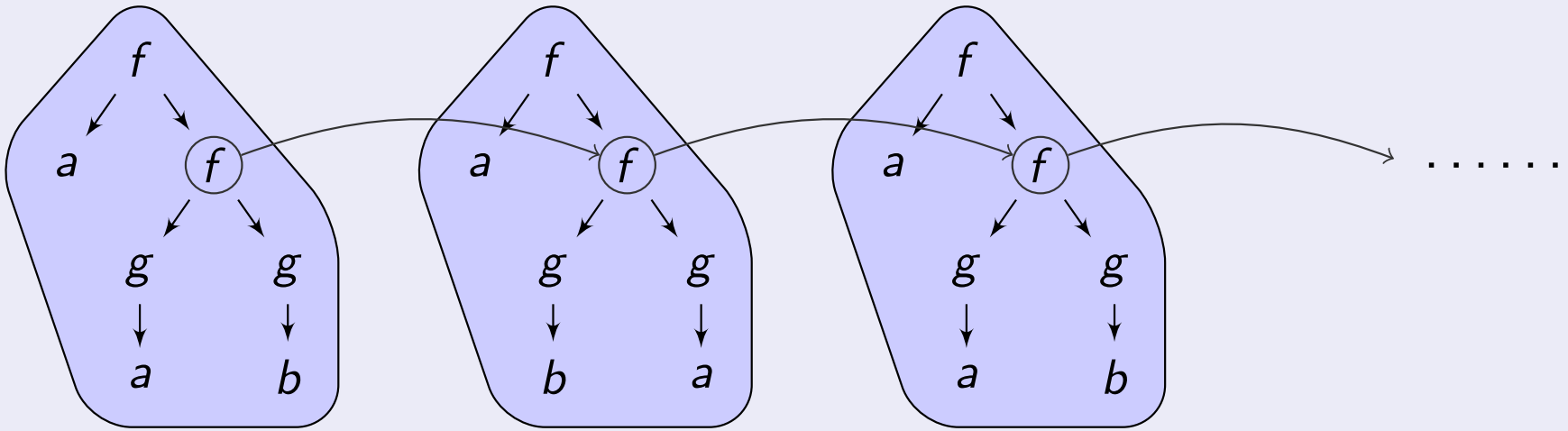
An Example

Reduction sequence for $f(x, y) \to f(y, x)$



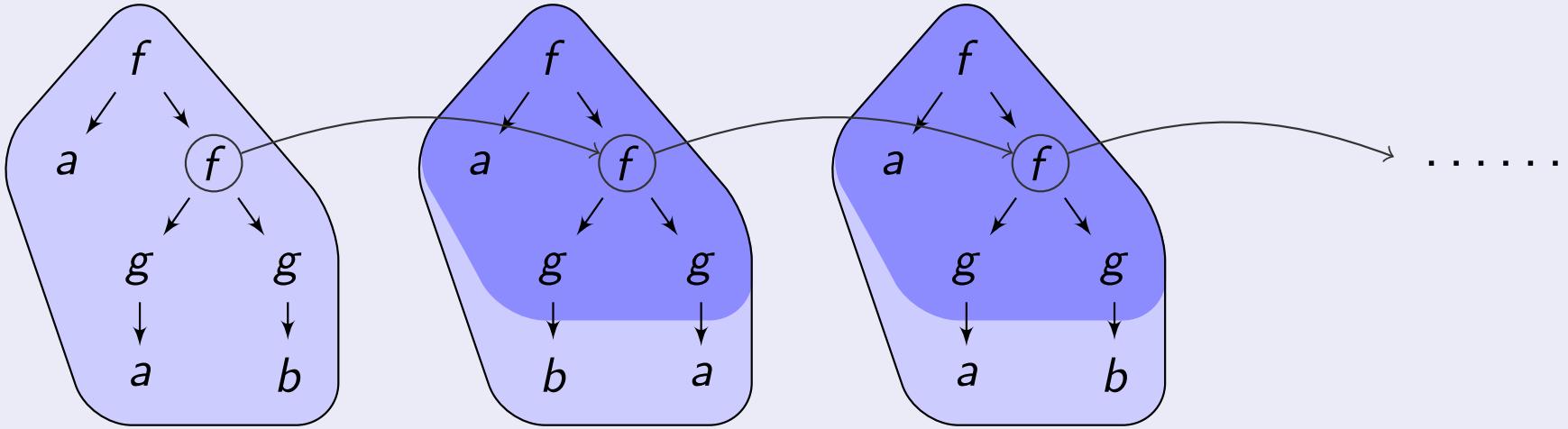
An Example

Reduction sequence for $f(x, y) \to f(y, x)$



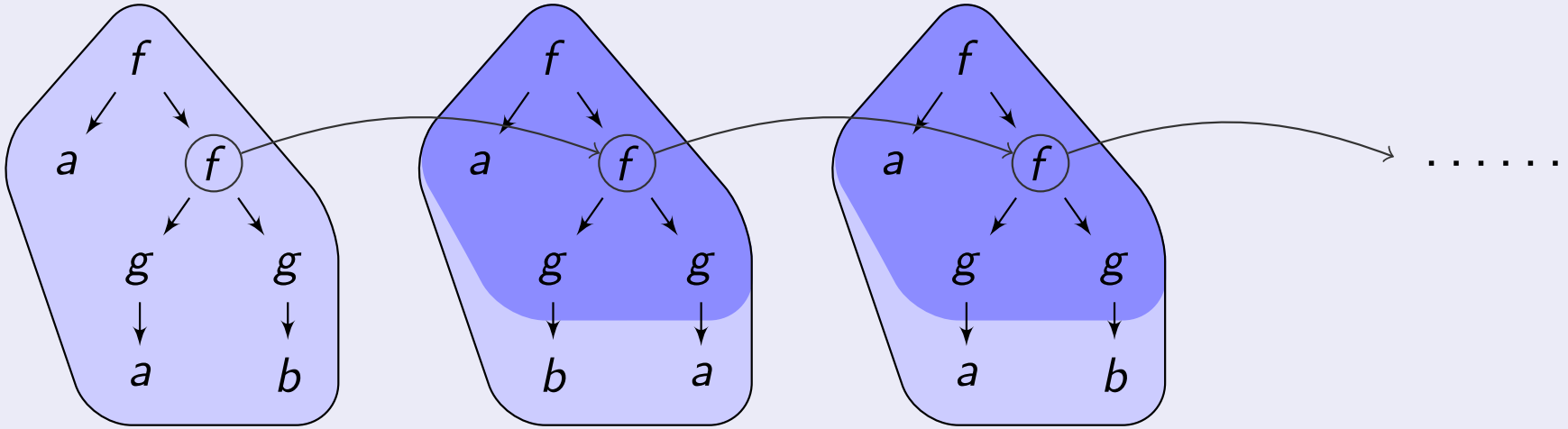
An Example

Reduction sequence for $f(x, y) \to f(y, x)$

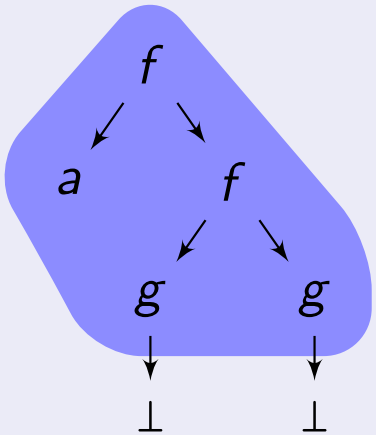


An Example

Reduction sequence for $f(x, y) \to f(y, x)$

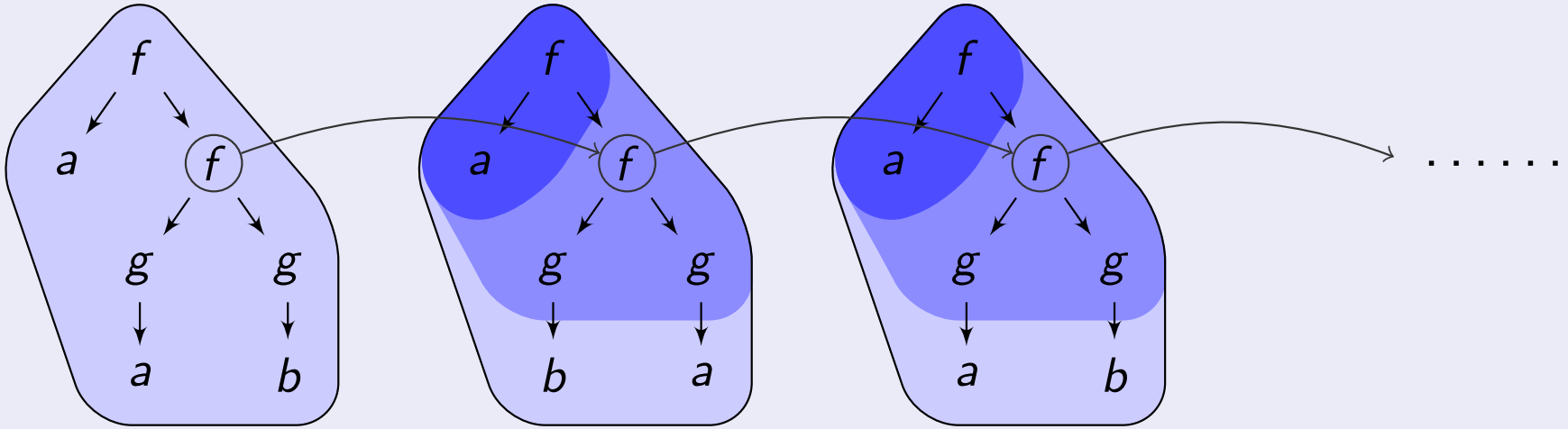


Weak convergence

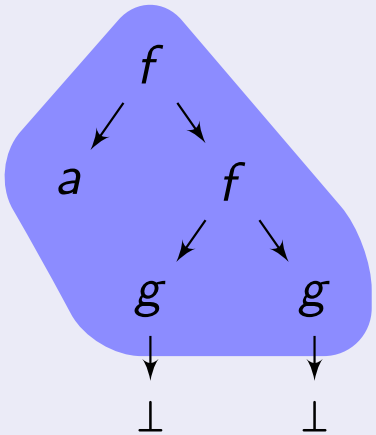


An Example

Reduction sequence for $f(x, y) \to f(y, x)$

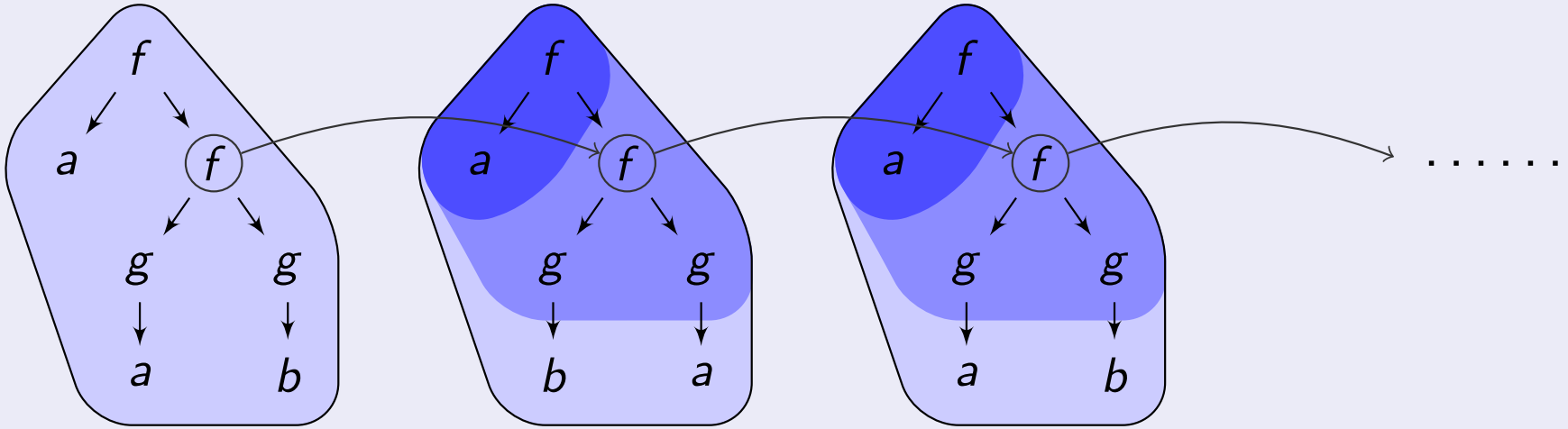


Weak convergence

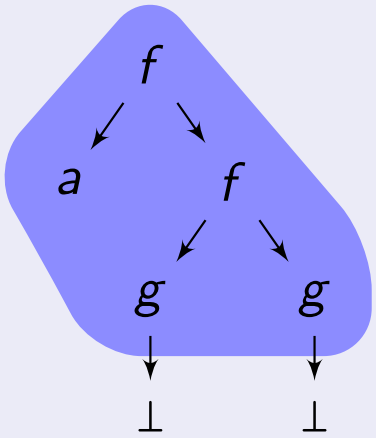


An Example

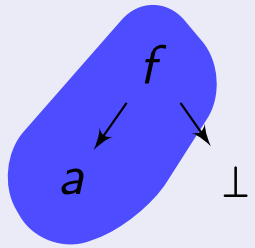
Reduction sequence for $f(x, y) \to f(y, x)$



Weak convergence



Strong convergence



Properties of the Partial Order Model

Benefits

- reduction sequences always converge for complete semilattices
- more fine-grained than the metric model
- more intuitive than the metric model
- under certain conditions (met by terms and term graphs) subsumes metric model

Partial Order Infinitary Term Rewriting

Properties of orthogonal systems

property	**metric**	**partial order**
compression		
finite approximation		
complete developments		
infinitary confluence		
infinitary normalisation		

Partial Order Infinitary Term Rewriting

Properties of orthogonal systems

property	**metric**	**partial order**
compression		
finite approximation		
complete developments		
infinitary confluence		
infinitary normalisation		

Compression

Every reduction can be performed in at most ω steps:

$$s \twoheadrightarrow^{\alpha} t \quad \Longrightarrow \quad s \twoheadrightarrow^{\leq \omega} t$$

Partial Order Infinitary Term Rewriting

Properties of orthogonal systems

property	metric	partial order
compression	✔	✔
finite approximation		
complete developments		
infinitary confluence		
infinitary normalisation		

Compression

Every reduction can be performed in at most ω steps:

$$s \twoheadrightarrow^{\alpha} t \quad \Longrightarrow \quad s \twoheadrightarrow^{\leq \omega} t$$

Partial Order Infinitary Term Rewriting

Properties of orthogonal systems

property	metric	partial order
compression	✔	✔
finite approximation		
complete developments		
infinitary confluence		
infinitary normalisation		

Finite approximation

Every outcome can be approximated by a finite reduction arbitrary well:

$$s \twoheadrightarrow^{\alpha} t \quad \Longrightarrow \quad \forall d \in \mathbb{N} \exists t' \begin{cases} s \to^{\star} t' \\ t \text{ and } t' \text{ coincide up to depth } d \end{cases}$$

Partial Order Infinitary Term Rewriting

Properties of orthogonal systems

property	**metric**	**partial order**
compression	✔	✔
finite approximation	✔	✔
complete developments		
infinitary confluence		
infinitary normalisation		

Finite approximation

Every outcome can be approximated by a finite reduction arbitrary well:

$$s \twoheadrightarrow^{\alpha} t \quad \Longrightarrow \quad \forall d \in \mathbb{N} \exists t' \begin{cases} s \to^{\star} t' \\ t \text{ and } t' \text{ coincide up to depth } d \end{cases}$$

Partial Order Infinitary Term Rewriting

Properties of orthogonal systems

property	metric	partial order
compression	✔	✔
finite approximation	✔	✔
complete developments		
infinitary confluence		
infinitary normalisation		

Complete developments

reductions simulating simultaneous contraction of a set of redexes

Partial Order Infinitary Term Rewriting

Properties of orthogonal systems

property	metric	partial order
compression	✔	✔
finite approximation	✔	✔
complete developments	✘	✔
infinitary confluence		
infinitary normalisation		

Complete developments

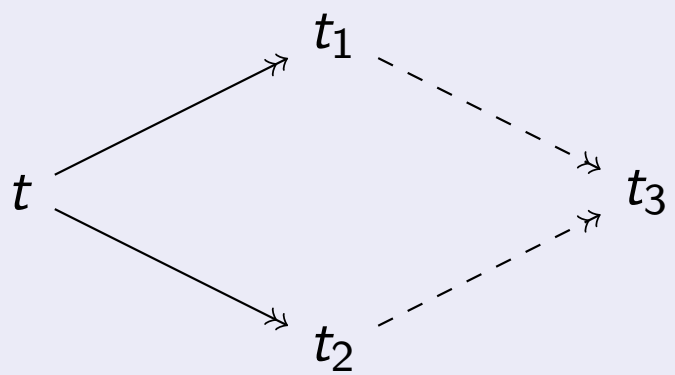
reductions simulating simultaneous contraction of a set of redexes

Partial Order Infinitary Term Rewriting

Properties of orthogonal systems

property	**metric**	**partial order**
compression	✔	✔
finite approximation	✔	✔
complete developments	✘	✔
infinitary confluence		
infinitary normalisation		

Infinitary confluence

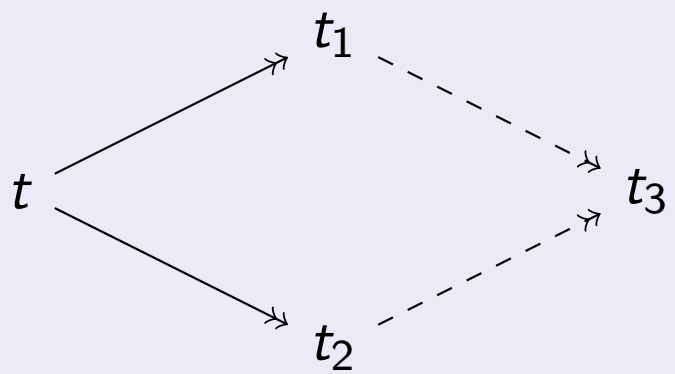


Partial Order Infinitary Term Rewriting

Properties of orthogonal systems

property	**metric**	**partial order**
compression	✔	✔
finite approximation	✔	✔
complete developments	✘	✔
infinitary confluence	✘	✔
infinitary normalisation		

Infinitary confluence



Partial Order Infinitary Term Rewriting

Properties of orthogonal systems

property	metric	partial order
compression	✔	✔
finite approximation	✔	✔
complete developments	✘	✔
infinitary confluence	✘	✔
infinitary normalisation		

Infinitary normalisation

every term has a normal form reachable by a possibly infinite reduction

Partial Order Infinitary Term Rewriting

Properties of orthogonal systems

property	metric	partial order
compression	✔	✔
finite approximation	✔	✔
complete developments	✘	✔
infinitary confluence	✘	✔
infinitary normalisation	✘	✔

Infinitary normalisation

every term has a normal form reachable by a possibly infinite reduction

Equivalence to Böhm Reduction

Definition (Böhm reduction)

▸ Skip Böhm Reduction ▸ Jump to Conclusions

Let $\mathcal{R}$ be a term rewriting system.

1. A term t is called a root-active if every reduct of t can be reduced to a redex.
2. The Böhm reduction $\mathcal{B}$ of $\mathcal{R}$ is the term rewriting system consisting of the rules

$$R \cup \{ t \to \bot \mid t \in \mathcal{RA}_\bot \setminus \{\bot\} \}$$

where $\mathcal{RA}_\bot$ is the set of root-active terms in which some root-active subterms are replaced by $\bot$.

Equivalence to Böhm Reduction

Definition (Böhm reduction)

Let $\mathcal{R}$ be a term rewriting system.

1. A term t is called a root-active if every reduct of t can be reduced to a redex.
2. The Böhm reduction $\mathcal{B}$ of $\mathcal{R}$ is the term rewriting system consisting of the rules

$$R \cup \{ t \to \bot \mid t \in \mathcal{RA}_\bot \setminus \{\bot\} \}$$

where $\mathcal{RA}_\bot$ is the set of root-active terms in which some root-active subterms are replaced by $\bot$.

Theorem (Böhm reductions = partial order reductions)

Let $\mathcal{R}$ be an orthogonal system.

1. *A total term t in $\mathcal{R}$ is root-active iff $t \twoheadrightarrow_\mathcal{R} \bot$.*
2. $s \twoheadrightarrow^p_\mathcal{R} t \quad \textit{iff} \quad s \twoheadrightarrow^m_\mathcal{B} t$

Term Graphs

What are term graphs?

▸ Skip Term Graphs

- generalisation of terms
- allow sharing of common substructures
- motivation: compact representation of terms

Term Graphs

What are term graphs?

- generalisation of terms
- allow sharing of common substructures
- motivation: compact representation of terms

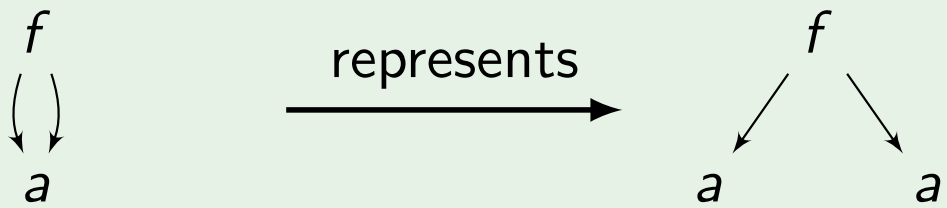
Example

Term Graphs

What are term graphs?

- generalisation of terms
- allow sharing of common substructures
- motivation: compact representation of terms

Example

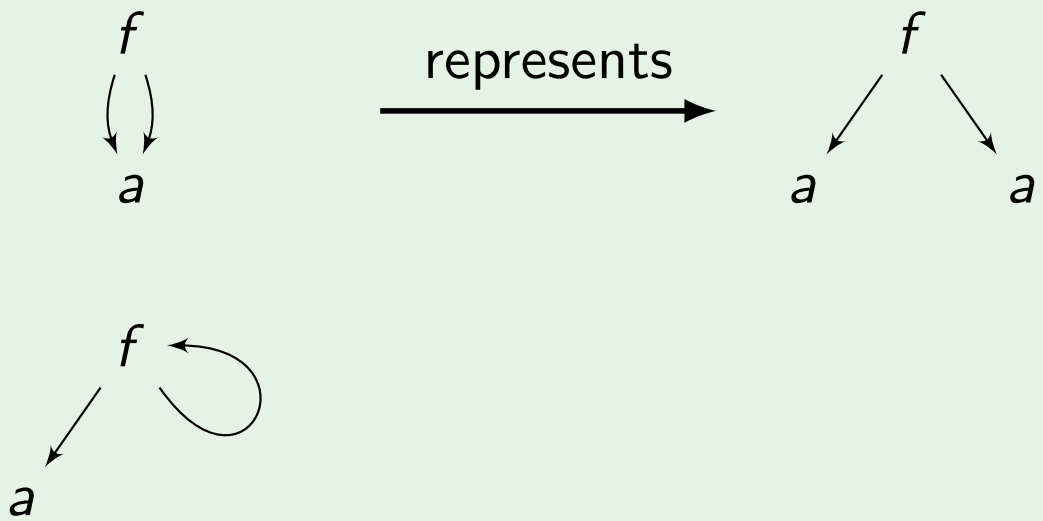


Term Graphs

What are term graphs?

- generalisation of terms
- allow sharing of common substructures
- motivation: compact representation of terms

Example

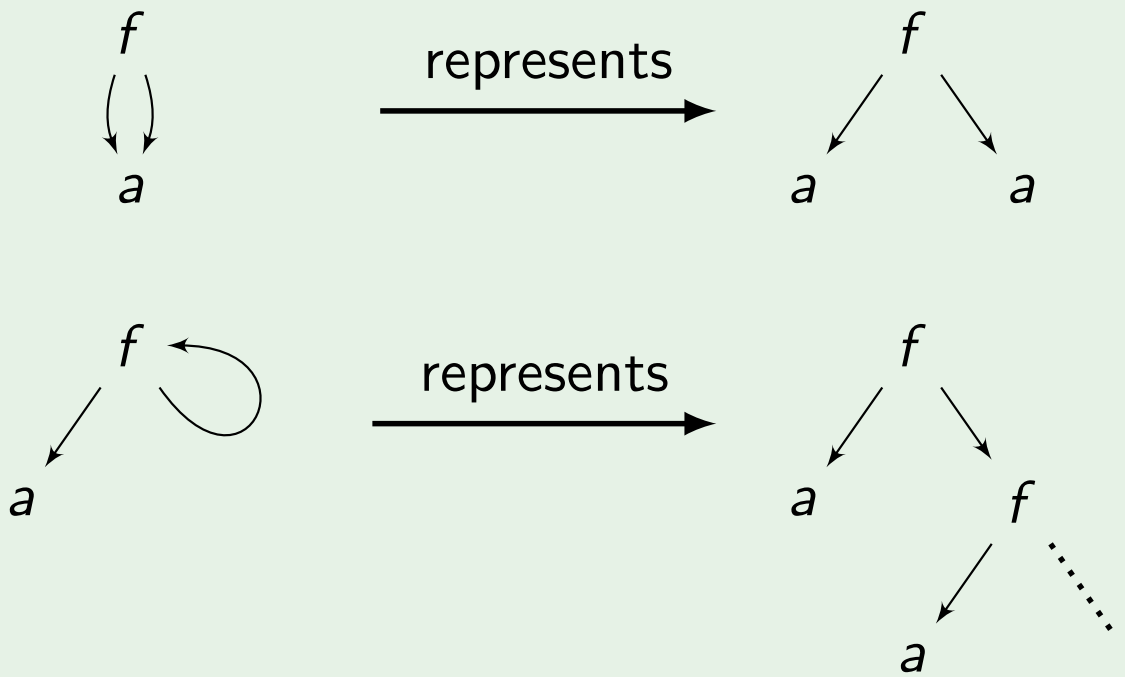


Term Graphs

What are term graphs?

- generalisation of terms
- allow sharing of common substructures
- motivation: compact representation of terms

Example



Term Graph Rewriting

Benefits

- non-cyclic term graph rewriting can be used to efficiently implement term rewriting
- cyclic term graph rewriting can be used to implement restricted forms of infinitary term rewriting (rational term rewriting)

Term Graph Rewriting

Benefits

- non-cyclic term graph rewriting can be used to efficiently implement term rewriting
- cyclic term graph rewriting can be used to implement restricted forms of infinitary term rewriting (rational term rewriting)

Introducing infinitary term graph rewriting

- complete ultrametric on terms is generalised to term graphs
- complete semilattice on terms is generalised to term graphs
- metric and partial order model of infinitary rewriting are applied to term graph rewriting

Term Graph Rewriting

Benefits

- non-cyclic term graph rewriting can be used to efficiently implement term rewriting
- cyclic term graph rewriting can be used to implement restricted forms of infinitary term rewriting (rational term rewriting)

Introducing infinitary term graph rewriting

- complete ultrametric on terms is generalised to term graphs
- complete semilattice on terms is generalised to term graphs
- metric and partial order model of infinitary rewriting are applied to term graph rewriting

Goal

identify closure properties of rational term rewriting

Outline

Results

Partial order model of infinitary rewriting

- more fine-grained model of convergence
- captures the intuition of convergence of reductions

Results

Partial order model of infinitary rewriting

- more fine-grained model of convergence
- captures the intuition of convergence of reductions

Infinitary term rewriting w.r.t. the partial order model

- subsumes the metric model
- infinitary normalising and confluent $\rightsquigarrow$ unique normal forms
- equivalent to Böhm reductions $\rightsquigarrow$ normal forms are Böhm trees

Results

Partial order model of infinitary rewriting

- more fine-grained model of convergence
- captures the intuition of convergence of reductions

Infinitary term rewriting w.r.t. the partial order model

- subsumes the metric model
- infinitary normalising and confluent $\rightsquigarrow$ unique normal forms
- equivalent to Böhm reductions $\rightsquigarrow$ normal forms are Böhm trees

Infinitary term graph rewriting

- a complete metric and a complete semilattice are devised
- infinitary term graph rewriting is introduced
- tool for investigating closure properties of rational term rewriting

Perspective

Further analysis of infinitary term graph rewriting

- generalise confluence results from finitary term graph rewriting
- identify which class of infinitary term rewriting can be simulated by (infinitary) term graph rewriting
- generalise Böhm trees (of terms) to "Böhm graphs" (of term graphs)

Perspective

Further analysis of infinitary term graph rewriting

- generalise confluence results from finitary term graph rewriting
- identify which class of infinitary term rewriting can be simulated by (infinitary) term graph rewriting
- generalise Böhm trees (of terms) to "Böhm graphs" (of term graphs)

Partial order infinitary term rewriting

- properties of weak convergence
- generalisation to higher-order term rewriting $\rightsquigarrow$ refine the partial order
- use other term graph rewriting approaches (equational and double-pushout approach) to simulate partial order infinitary term rewriting

Perspective (contd.)

Implementing infinitary term rewriting

- find closure properties of rational term rewriting
- find more heuristics to transform term rewriting systems to term graph rewriting systems in order to implement infinitary term rewriting

Perspective (contd.)

Implementing infinitary term rewriting

- find closure properties of rational term rewriting
- find more heuristics to transform term rewriting systems to term graph rewriting systems in order to implement infinitary term rewriting

Applications to functional programming

- reason about lazy functional programs
- optimisation through cyclic term graph rewriting

References

Nachum Dershowitz, Stéphane Kaplan, and David A. Plaisted.
Rewrite, rewrite, rewrite, rewrite, rewrite, ...
Theoretical Computer Science, 83(1):71–96, 1991.

Richard Kennaway, Jan Willem Klop, M. Ronan Sleep, and Fer-Jan de Vries.
Transfinite reductions in orthogonal term rewriting systems.
Information and Computation, 119(1):18–38, 1995.

Richard Kennaway.
On transfinite abstract reduction systems.
Technical report, CWI, Amsterdam, 1992.

Richard Kennaway, Jan Willem Klop, M. Ronan Sleep, and Fer-Jan de Vries.
On the adequacy of graph rewriting for simulating term rewriting.
ACM Transactions on Programming Languages and Systems, 16(3):493–523, 1994.

References (contd.)


Richard Kennaway, Vincent van Oostrom, and Fer-Jan de Vries.
Meaningless terms in rewriting.
Journal of Functional and Logic Programming, 1999(1):1–35, February 1999.


Jeroen Ketema.
Böhm-Like Trees for Rewriting.
PhD thesis, Vrije Universiteit Amsterdam, 2006.


Stefan Blom.
An approximation based approach to infinitary lambda calculi.
Rewriting Techniques and Applications, RTA, 2004.