Nested Term Graphs

(work in progress)

Clemens Grabmayer and Vincent van Oostrom

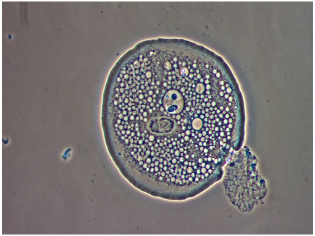
Computer Science (VU University Amsterdam) and Philosophy (Utrecht University)

TERMGRAPH 2014

13 July 2014

nested

‘a group of objects made to fit close together or one within another’



$$x = \sqrt{2 + \sqrt{2 + \sqrt{2 + \sqrt{2 + \ldots}}}}$$

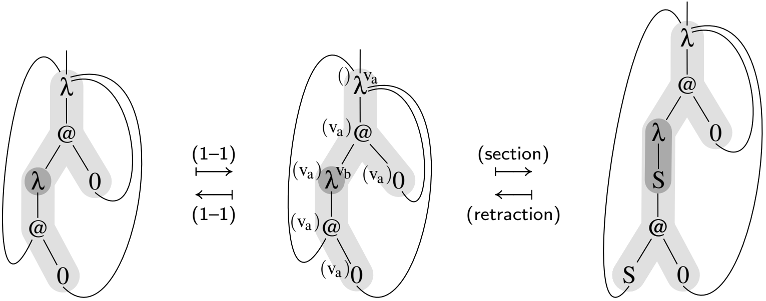
```
for i = 0 to 9 do
   for j = 0 to 9 do
     for k = 0 to 9 do
       sum = sum + i*100
                   + j*10 + k + 1;
```

nested term graphs

- motivation
  - an implementation of higher-order term graphs
    as first-order term graphs
  - representing nested scope structure of terms in $\boldsymbol{\lambda}$ or in $\boldsymbol{\lambda}_{\text{letrec}}$
- definitions
  - intensional definition as: recursive graph specifications
  - extensional definition as: enriched first-order term graphs
- bisimulation, and nested bisimulation
- implementation as first-order term graphs
- further investigations and aims

higher-order as first-order term graphs [TERMGRAPH 2013]

let $f = \lambda x.(\lambda y.f\ x)x$ in f



higher-order term graph [Blom '03]

higher-order term graph (abstraction-prefix funct.)

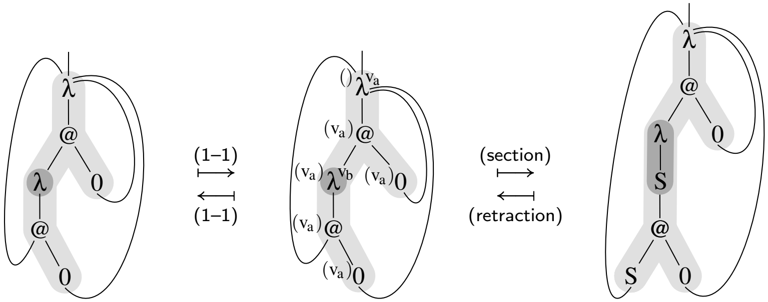
first-order term graph

CG, Jan Rochel:

- *Term Graph Representations for Cyclic Lambda Terms*, TG 2013.
- *Maximal Sharing in the Lambda Calculus with Letrec*, ICFP 2014.

higher-order as first-order term graphs [TERMGRAPH 2013]

$$\text{let } f = \lambda x.(\lambda y.f\ x)x \text{ in } f$$

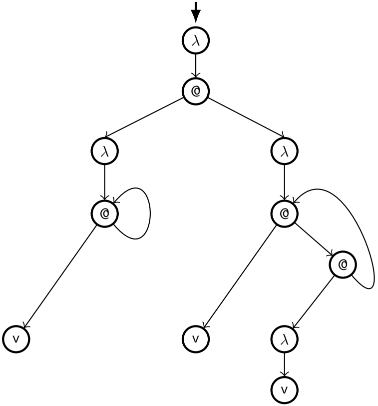


higher-order term graph [Blom '03] | higher-order term graph (abstraction-prefix funct.) | first-order term graph

CG, Jan Rochel:

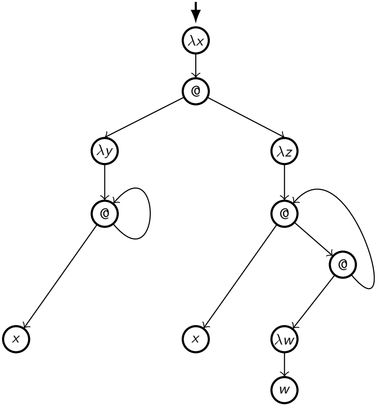
- *Term Graph Representations for Cyclic Lambda Terms*, TG 2013.
- *Maximal Sharing in the Lambda Calculus with Letrec*, ICFP 2014.

nested scopes in λ-terms



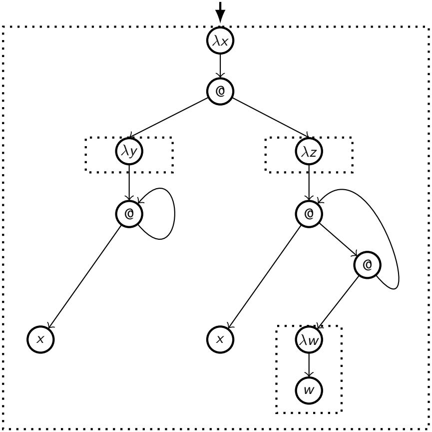
first-order term graph over $\Sigma = \{\lambda/1,\ @/2,\ \mathsf{v}/0\}$

nested scopes in λ-terms



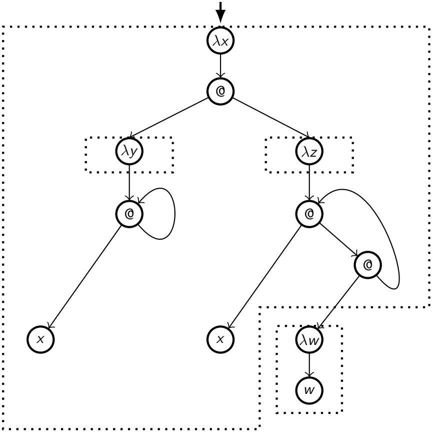
$$\lambda x.(\lambda y.\text{let } \alpha = x\alpha \text{ in } \alpha)(\lambda z.\text{let } \beta = x(\lambda w.w)\beta \text{ in } \beta)$$

nested scopes in λ-terms



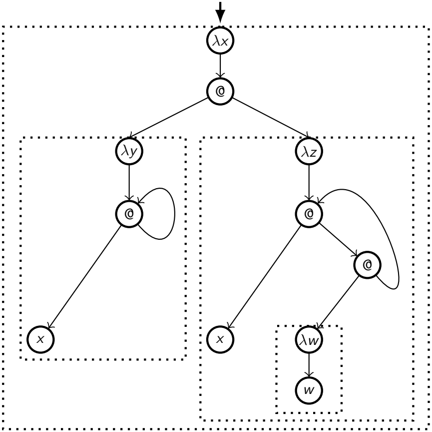
$$\lambda x.(\lambda y.\text{let } \alpha = x\alpha \text{ in } \alpha)(\lambda z.\text{let } \beta = x(\lambda w.w)\beta \text{ in } \beta)$$

nested scopes in λ-terms



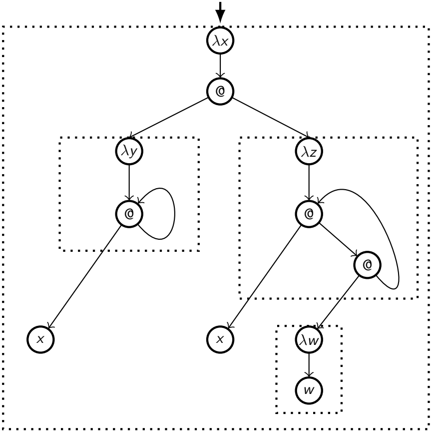
$$\lambda x.(\lambda y.\mathrm{let}\ \alpha = x\alpha\ \mathrm{in}\ \alpha)(\lambda z.\mathrm{let}\ \beta = x(\lambda w.w)\beta\ \mathrm{in}\ \beta)$$

nested scopes in λ-terms



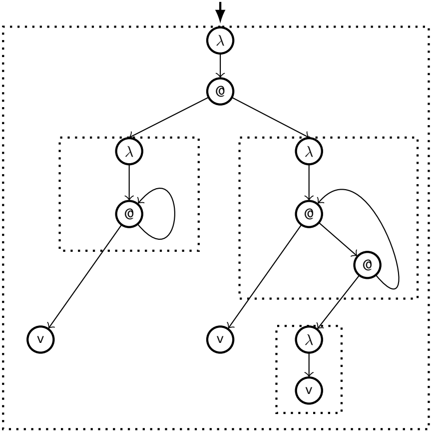
$$\lambda x.(\lambda y.\text{let } \alpha = x\alpha \text{ in } \alpha)(\lambda z.\text{let } \beta = x(\lambda w.w)\beta \text{ in } \beta)$$

nested scopes in λ-terms



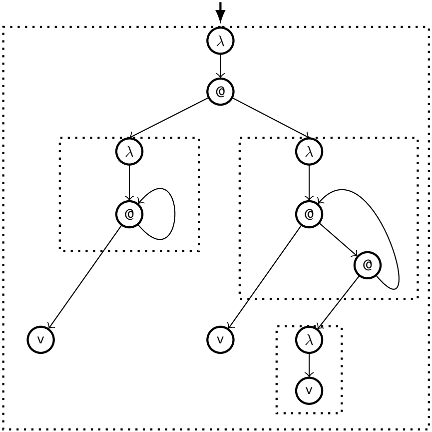
$$\lambda x.(\lambda y.\text{let } \alpha = x\alpha \text{ in } \alpha)(\lambda z.\text{let } \beta = x(\lambda w.w)\beta \text{ in } \beta)$$

nested scopes in λ-terms

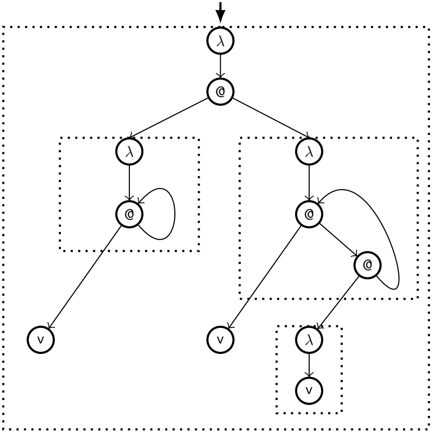


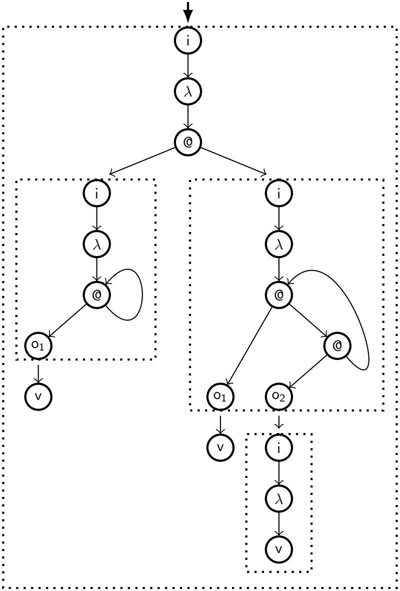
$$\lambda x.(\lambda y.\text{let } \alpha = x\alpha \text{ in } \alpha)(\lambda z.\text{let } \beta = x(\lambda w.w)\beta \text{ in } \beta)$$

nested scopes in λ-terms



nested scopes $\rightarrowtriangle$ nested term graph





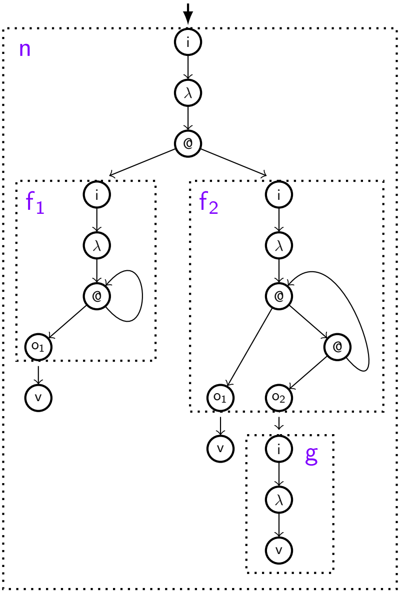
nested term graph

gletrec

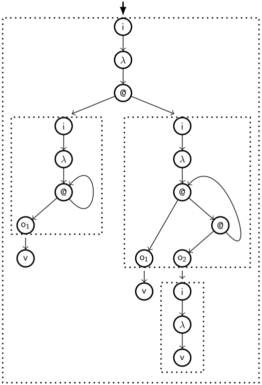
$$\begin{array}{rcl} \mathsf{n}() & = & \lambda x.\mathsf{f}_1(x)\mathsf{f}_2(x,\mathsf{g}()) \\ \mathsf{f}_1(X_1) & = & \lambda x.\mathrm{let}\,\alpha = X_1\alpha\,\mathrm{in}\,\alpha \\ \mathsf{f}_2(X_1,X_2) & = & \lambda y.\mathrm{let}\,\beta = X_1(X_2\beta)\,\mathrm{in}\,\beta \\ \mathsf{g}() & = & \lambda z.z \end{array}$$

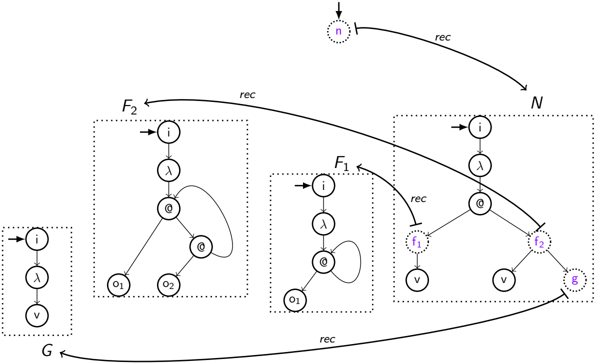
in

$$\mathsf{n}()$$



nested term graph





signature

A *signature for nested term graphs* (*ntg-signature*) is a signature Σ that is partitioned into:

- *atomic* symbol alphabet Σ_{at}
- *nested* symbol alphabet Σ_{ne}

Additionally used:

- *interface* symbols alphabet $IO = I \cup O$
  - $I = \{\mathsf{i}\}$ with i unary
  - $O = \{\mathsf{o}_1, \mathsf{o}_2, \mathsf{o}_3, \ldots\}$ with o_i nullary

recursive graph specification

Definition

Let Σ be an ntg-signature.
A *recursive graph specification* (a *rgs*) $\mathcal{R} = \langle rec, r \rangle$ consists of:

- *specification function*

$$rec : \Sigma_{\mathrm{ne}} \longrightarrow \mathrm{TG}(\Sigma \cup IO)$$
$$f_{/k} \longmapsto rec(f) = F \in \mathrm{TG}(\Sigma \cup \{\mathrm{i}, \mathrm{o}_1, \ldots, \mathrm{o}_k\})$$

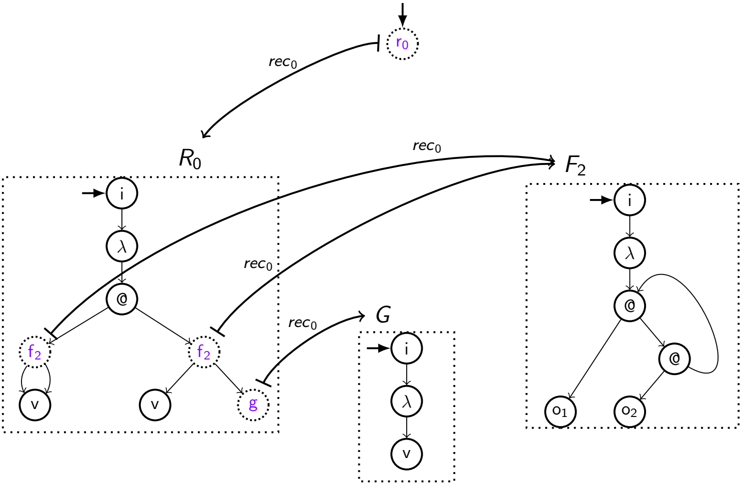
  where F contains precisely one vertex labeled by i, the root, and one vertex each labeled by o_i, for $i \in \{1, \ldots, k\}$;
- nullary *root symbol* $r \in \Sigma_{\mathrm{ne}}$.

rooted *dependency* ARS $\multimap$ of $\mathcal{R}$:

- objects: nested symbols in Σ_{ne}
- steps: for all $f, g \in \Sigma_{\mathrm{ne}}$:

$$p : f \multimap g \iff g \text{ occurs in the term graph } rec(f) \text{ at position } p$$

recursive graph specification



$$\Sigma_{at} = \{\lambda/1,\ @/2,\ v/0\},\ \Sigma_{ne} = \{r_0/0,\ f_2/2,\ g/0\},\ I = \{i/1\},\ O = \{o_1/0, o_2/0, \ldots\}.$$

recursive graph specification

Definition

Let Σ be an ntg-signature.
A *recursive graph specification* (a *rgs*) $\mathcal{R} = \langle rec, r \rangle$ consists of:

- *specification function*

$$rec : \Sigma_{\mathrm{ne}} \longrightarrow \mathrm{TG}(\Sigma \cup IO)$$
$$f/k \longmapsto rec(f) = F \in \mathrm{TG}(\Sigma \cup \{\mathrm{i}, \mathrm{o}_1, \ldots, \mathrm{o}_k\})$$

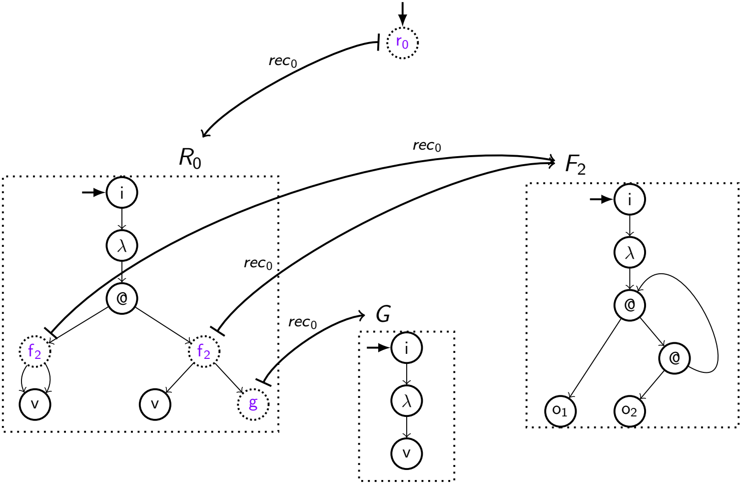
  where F contains precisely one vertex labeled by i, the root, and one vertex each labeled by o_i, for $i \in \{1, \ldots, k\}$;
- nullary *root symbol* $r \in \Sigma_{\mathrm{ne}}$.

rooted *dependency* ARS $\multimap$ of $\mathcal{R}$:

- objects: nested symbols in Σ_{ne}
- steps: for all $f, g \in \Sigma_{\mathrm{ne}}$:

$$p : f \multimap g \iff g \text{ occurs in the term graph } rec(f) \text{ at position } p$$

recursive graph specification



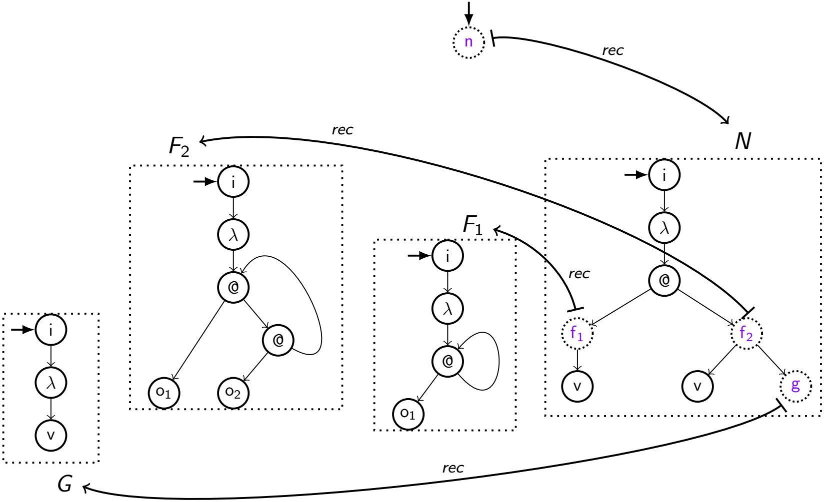
dependency ARS: $f_2 \rightrightarrows r_0 \leftarrow g$ is a dag (but not a tree).

nested term graph: intensional definition

Definition

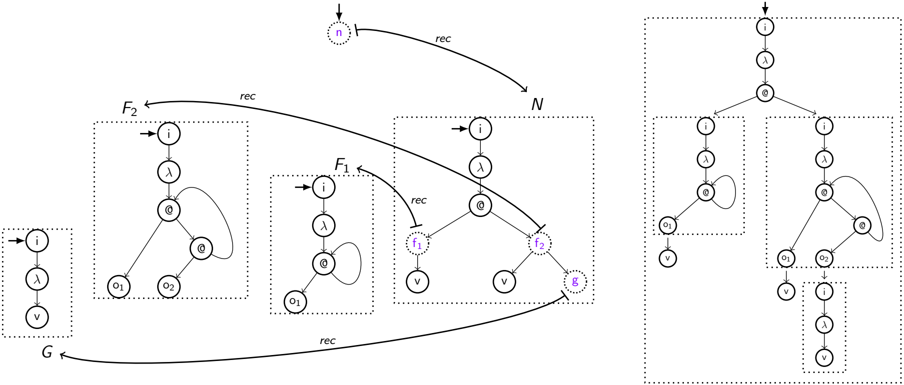
Let Σ be an ntg-signature.
A *nested term graph* over Σ is an rgs $\mathcal{N} = \langle rec, r \rangle$ such that the rooted dependency ARS $\multimap$ is a tree.

nested term graph (intensionally)



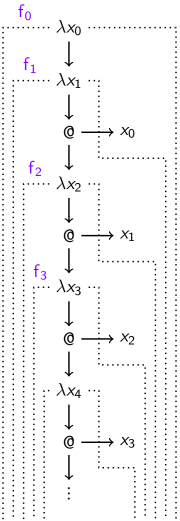
dependency ARS: $f_1 \multimap n \begin{array}{l} \multimapinv f_2 \\ \multimapinv g \end{array}$ is a tree.

nested term graph (intensionally)



dependency ARS: $\mathsf{f}_1 \multimap \mathsf{n} \begin{array}{l} \multimap\ \mathsf{f}_2 \\ \multimap\ \mathsf{g} \end{array}$ is a tree.

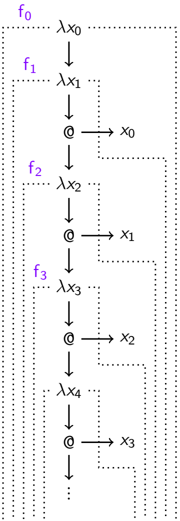
nested term graph (intensionally)



infinite λ-term
(infinitely nested scopes)

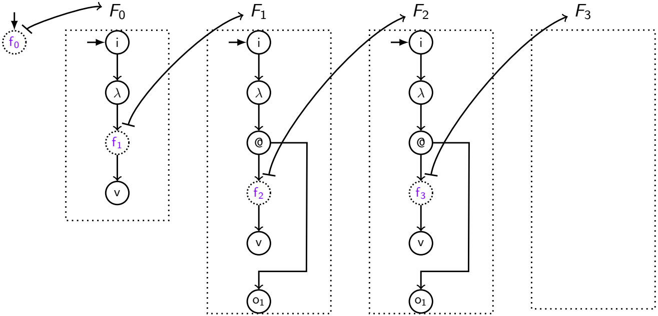
nested term graph with infinite nesting
dependency ARS: $f_0 ⟜ f_1 ⟜ f_2 ⟜ f_3 ⟜ \ldots$

nested term graph (intensionally)



infinite λ-term

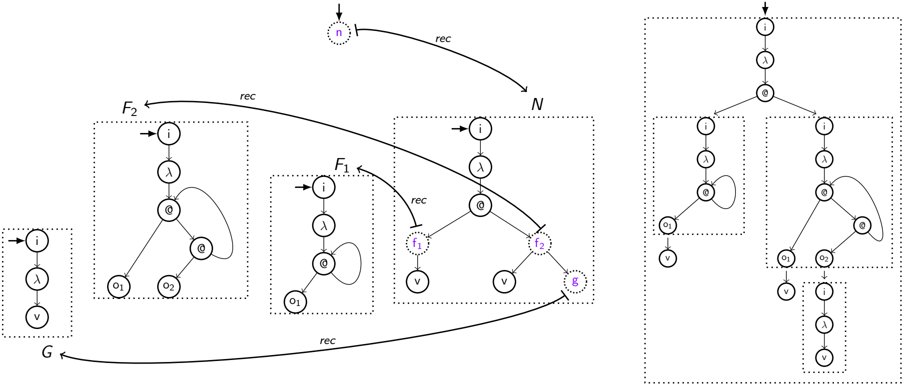
(infinitely nested scopes)



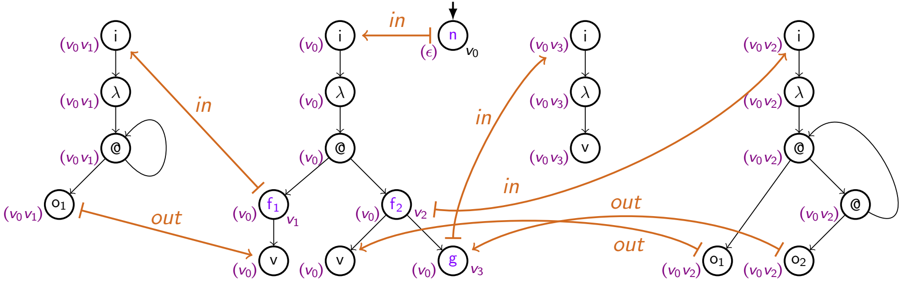
nested term graph with infinite nesting

dependency ARS: $\mathsf{f}_0 \multimapinv \mathsf{f}_1 \multimapinv \mathsf{f}_2 \multimapinv \mathsf{f}_3 \multimapinv \ldots$

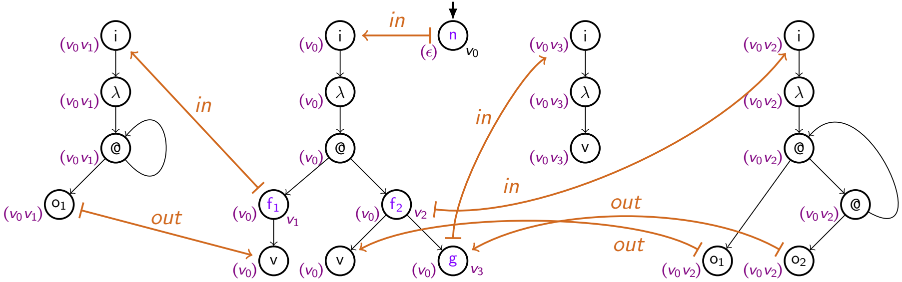
nested term graph (intensionally)



nested term graph: extensional definition



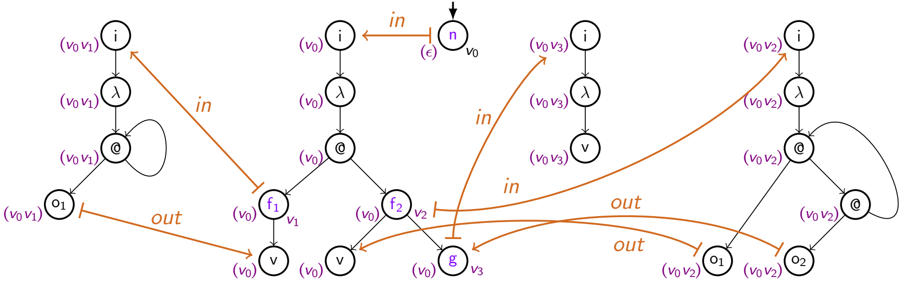
nested term graph: extensional definition



An *extensional description* of an ntg (an *entg*) over Σ is a term graph over $\Sigma \cup IO$ with vertex set V enriched by:

- *in* : $V \rightharpoonup V$, (v with nested symbol) $\mapsto$ (root of graph nested into v)

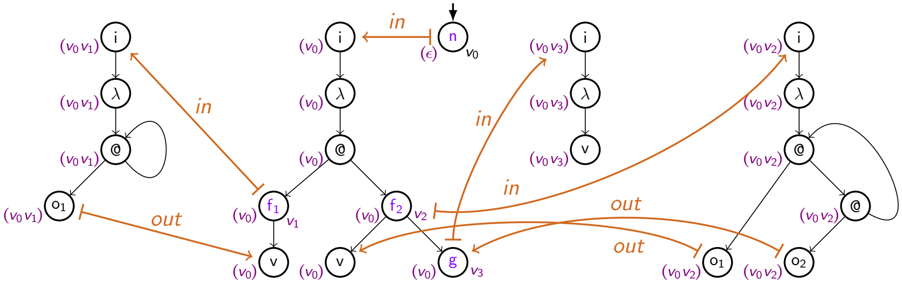
nested term graph: extensional definition



An *extensional description* of an ntg (an *entg*) over Σ is a term graph over $\Sigma \cup IO$ with vertex set V enriched by:

- *in* : $V \rightharpoonup V$, (v with nested symbol) $\mapsto$ (root of graph nested into v)
- *out* : $V \rightharpoonup V$, (v with output vertex o_i) $\mapsto$ (i-th successor of vertex into which the graph containing v is nested)

nested term graph: extensional definition



An *extensional description* of an ntg (an *entg*) over Σ is a term graph over $\Sigma \cup IO$ with vertex set V enriched by:

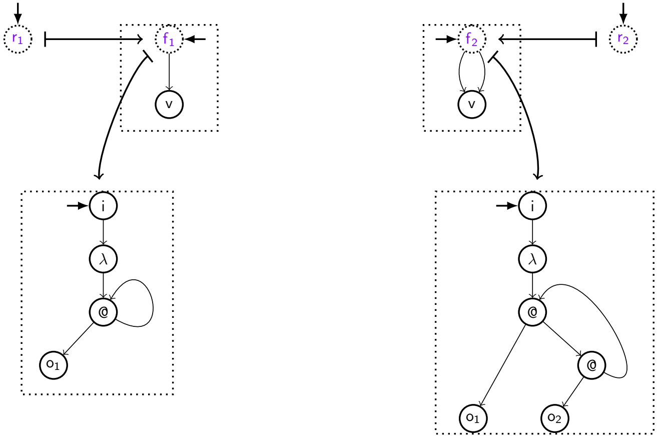
- *in* : $V \rightharpoonup V$, (v with nested symbol) $\mapsto$ (root of graph nested into v)
- *out* : $V \rightharpoonup V$, (v with output vertex o_i) $\mapsto$ (i-th successor of vertex into which the graph containing v is nested)
- *anc* : $V \to V^*$ *ancestor function*:
  $v \mapsto$ word $anc(v) = v_1 \cdots v_n$ of the vertices in which v is nested

nested term graphs: intensional vs. extensional definition

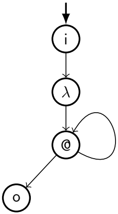
Proposition

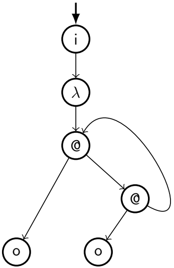
- *Every nested term graph has an extensional description.*
- *For every entg $\mathcal{G}$ there is a nested term graph for which $\mathcal{G}$ is the extensional description.*

bisimulation

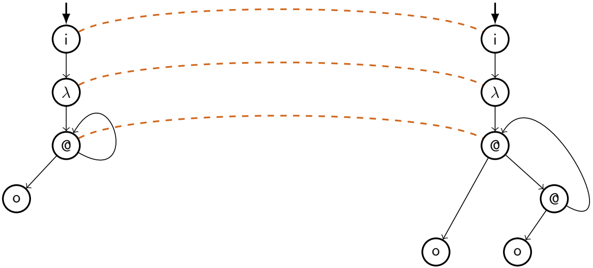


bisimulation between f-o term graphs



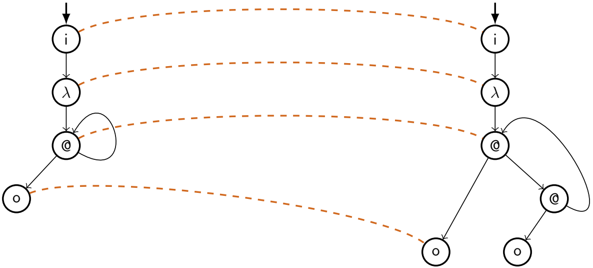


bisimulation between f-o term graphs



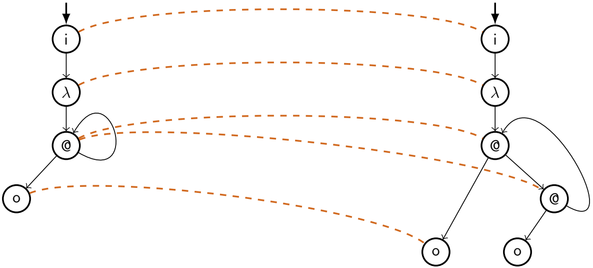
progression condition: *i*-th successors of related vertices must be related

bisimulation between f-o term graphs



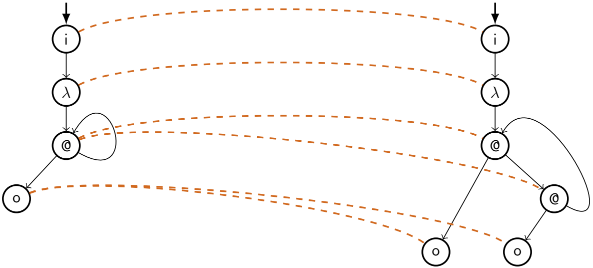
progression condition: i-th successors of related vertices must be related

bisimulation between f-o term graphs



progression condition: *i*-th successors of related vertices must be related

bisimulation between f-o term graphs



progression condition: *i*-th successors of related vertices must be related

bisimulation (for intensional ntg-definition)

Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be nested term graphs. Let V_1 the disjoint union of the vertices of term graphs in $\mathcal{N}_1$. Similar for V_2 w.r.t. $\mathcal{N}_2$.

bisimulation (for intensional ntg-definition)

Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be nested term graphs. Let V_1 the disjoint union of the vertices of term graphs in $\mathcal{N}_1$. Similar for V_2 w.r.t. $\mathcal{N}_2$.

$\mathcal{N}_1$ and $\mathcal{N}_2$ are bisimilar (denoted by $\mathcal{N}_1 \underline{\leftrightarrow} \mathcal{N}_2$) if there is bisimulation between $\mathcal{N}_1$ and $\mathcal{N}_2$, i.e. a binary relation ϕ betw. V_1 and V_2 such that:

- roots are related
- related vertices either <u>both</u> have nested labels, of <u>both</u> have interface labels, or <u>both</u> have <u>the same</u> atomic label

bisimulation (for intensional ntg-definition)

Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be nested term graphs. Let V_1 the disjoint union of the vertices of term graphs in $\mathcal{N}_1$. Similar for V_2 w.r.t. $\mathcal{N}_2$.

$\mathcal{N}_1$ and $\mathcal{N}_2$ are bisimilar (denoted by $\mathcal{N}_1 \underline{\leftrightarrow} \mathcal{N}_2$) if there is bisimulation between $\mathcal{N}_1$ and $\mathcal{N}_2$, i.e. a binary relation ϕ betw. V_1 and V_2 such that:

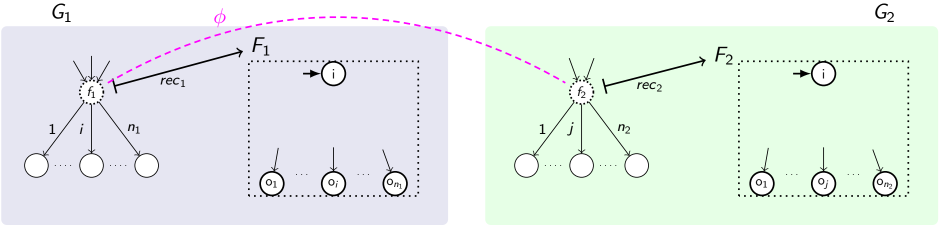
- roots are related
- related vertices either <u>both</u> have nested labels, of <u>both</u> have interface labels, or <u>both</u> have <u>the same</u> atomic label
- progression on <u>atomic</u> vertices: as for f-o term graphs

bisimulation (for intensional ntg-definition)

Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be nested term graphs. Let V_1 the disjoint union of the vertices of term graphs in $\mathcal{N}_1$. Similar for V_2 w.r.t. $\mathcal{N}_2$.

$\mathcal{N}_1$ and $\mathcal{N}_2$ are bisimilar (denoted by $\mathcal{N}_1 \leftrightarrow \mathcal{N}_2$) if there is bisimulation between $\mathcal{N}_1$ and $\mathcal{N}_2$, i.e. a binary relation ϕ betw. V_1 and V_2 such that:

- roots are related
- related vertices either both have nested labels, of both have interface labels, or both have the same atomic label
- progression on atomic vertices: as for f-o term graphs
- progression on nested vertices: interface clause

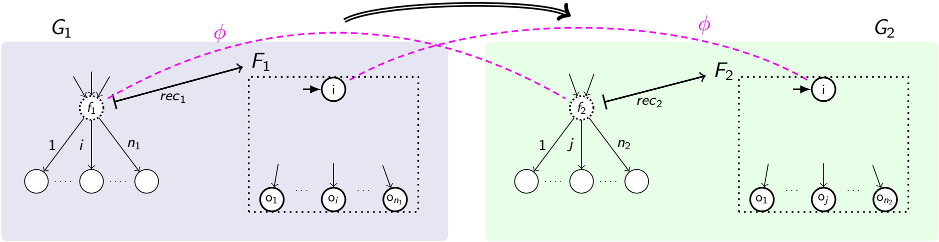


bisimulation (for intensional ntg-definition)

Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be nested term graphs. Let V_1 the disjoint union of the vertices of term graphs in $\mathcal{N}_1$. Similar for V_2 w.r.t. $\mathcal{N}_2$.

$\mathcal{N}_1$ and $\mathcal{N}_2$ are bisimilar (denoted by $\mathcal{N}_1 \underline{\leftrightarrow} \mathcal{N}_2$) if there is bisimulation between $\mathcal{N}_1$ and $\mathcal{N}_2$, i.e. a binary relation ϕ betw. V_1 and V_2 such that:

- roots are related
- related vertices either both have nested labels, of both have interface labels, or both have the same atomic label
- progression on atomic vertices: as for f-o term graphs
- progression on nested vertices: interface clause

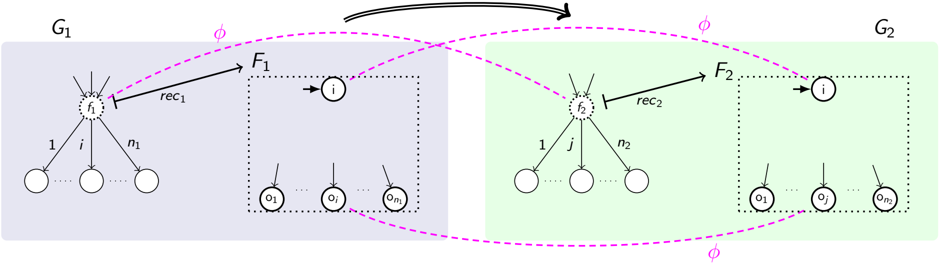


bisimulation (for intensional ntg-definition)

Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be nested term graphs. Let V_1 the disjoint union of the vertices of term graphs in $\mathcal{N}_1$. Similar for V_2 w.r.t. $\mathcal{N}_2$.

$\mathcal{N}_1$ and $\mathcal{N}_2$ are bisimilar (denoted by $\mathcal{N}_1 \underline{\leftrightarrow} \mathcal{N}_2$) if there is bisimulation between $\mathcal{N}_1$ and $\mathcal{N}_2$, i.e. a binary relation ϕ betw. V_1 and V_2 such that:

- roots are related
- related vertices either <u>both</u> have nested labels, of <u>both</u> have interface labels, or <u>both</u> have <u>the same</u> atomic label
- progression on <u>atomic</u> vertices: as for f-o term graphs
- progression on <u>nested</u> vertices: interface clause

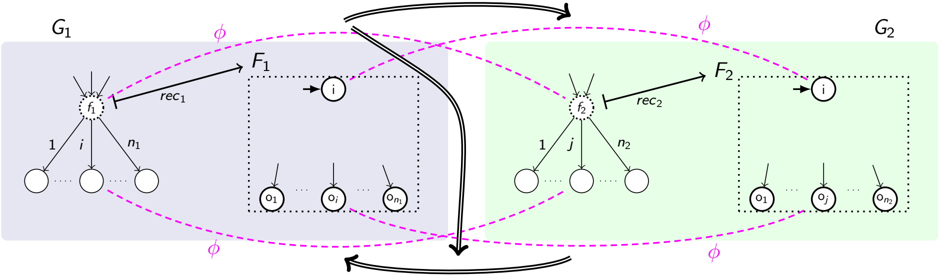


bisimulation (for intensional ntg-definition)

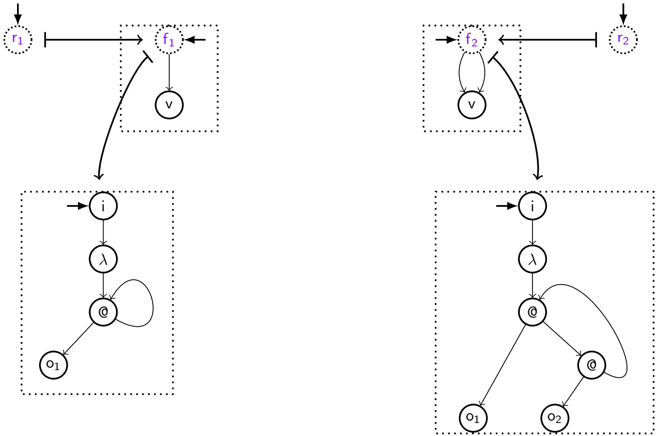
Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be nested term graphs. Let V_1 the disjoint union of the vertices of term graphs in $\mathcal{N}_1$. Similar for V_2 w.r.t. $\mathcal{N}_2$.

$\mathcal{N}_1$ and $\mathcal{N}_2$ are bisimilar (denoted by $\mathcal{N}_1 \leftrightarrow \mathcal{N}_2$) if there is bisimulation between $\mathcal{N}_1$ and $\mathcal{N}_2$, i.e. a binary relation ϕ betw. V_1 and V_2 such that:

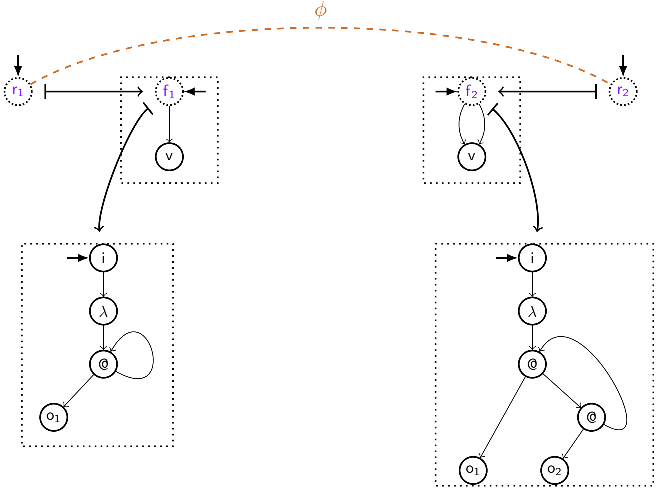
- roots are related
- related vertices either both have nested labels, of both have interface labels, or both have the same atomic label
- progression on atomic vertices: as for f-o term graphs
- progression on nested vertices: interface clause



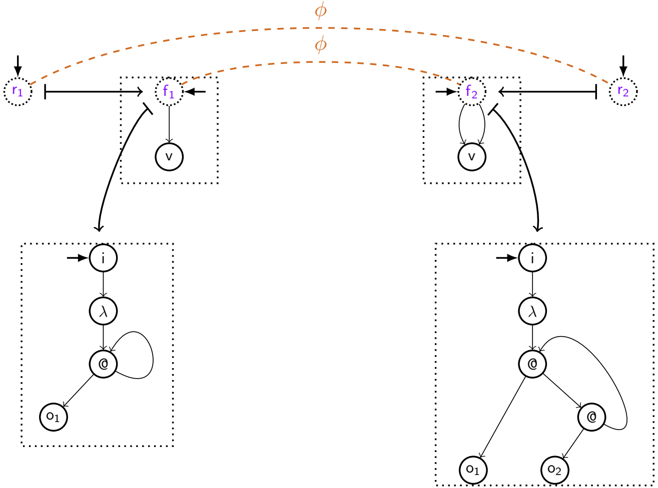
bisimulation (for intensional ntg-definition)



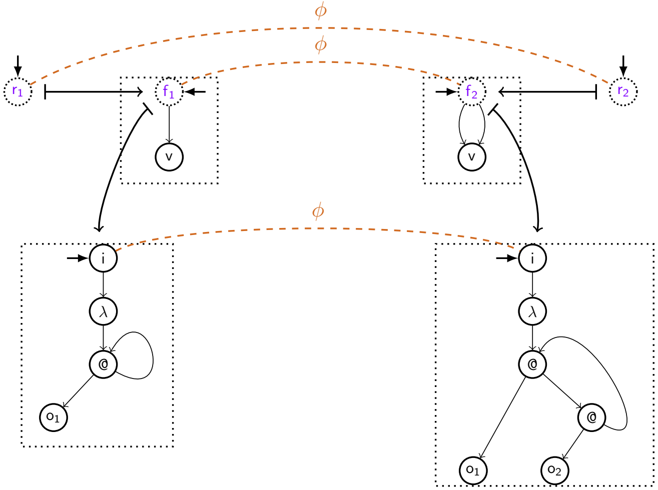
bisimulation (for intensional ntg-definition)



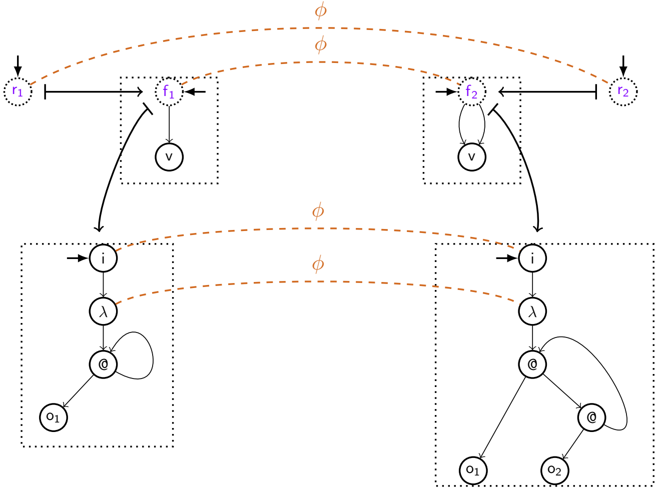
bisimulation (for intensional ntg-definition)



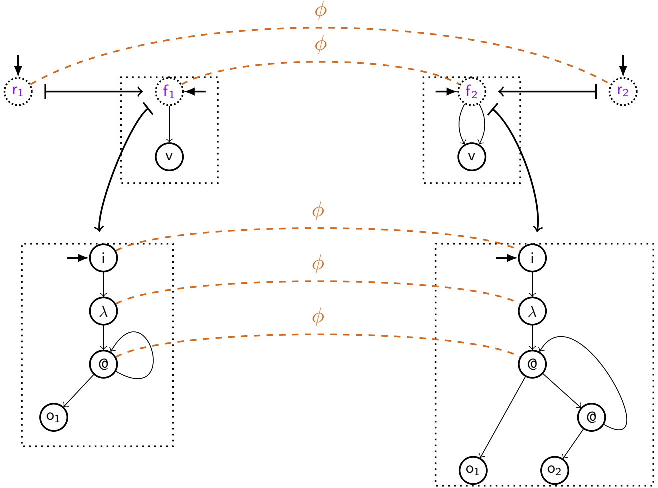
bisimulation (for intensional ntg-definition)



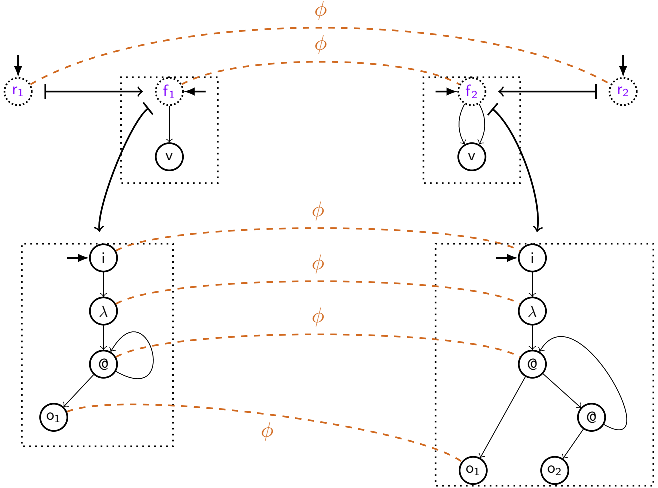
bisimulation (for intensional ntg-definition)



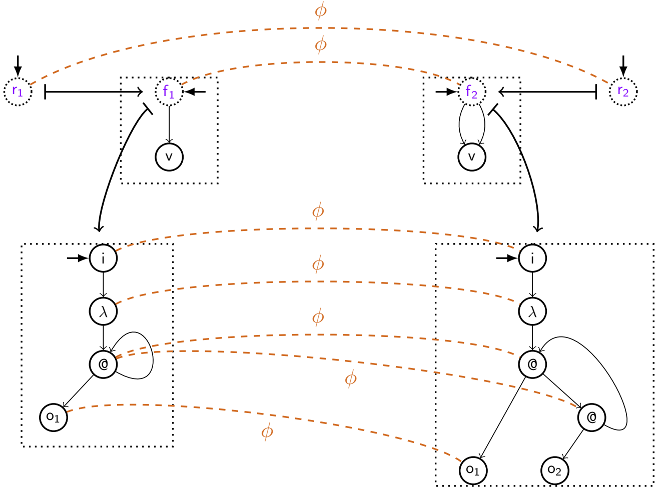
bisimulation (for intensional ntg-definition)



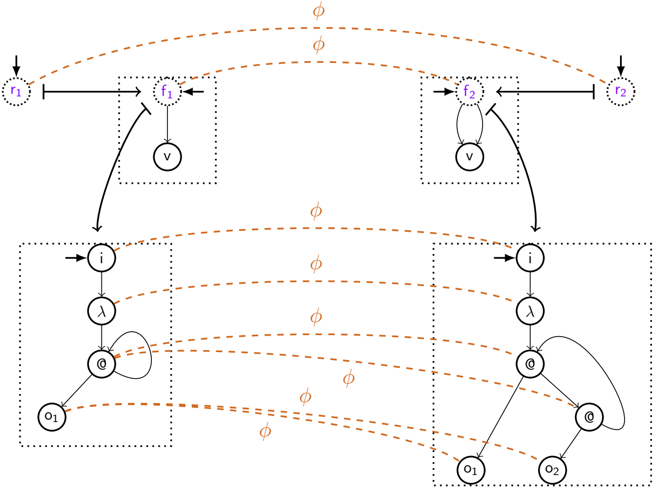
bisimulation (for intensional ntg-definition)



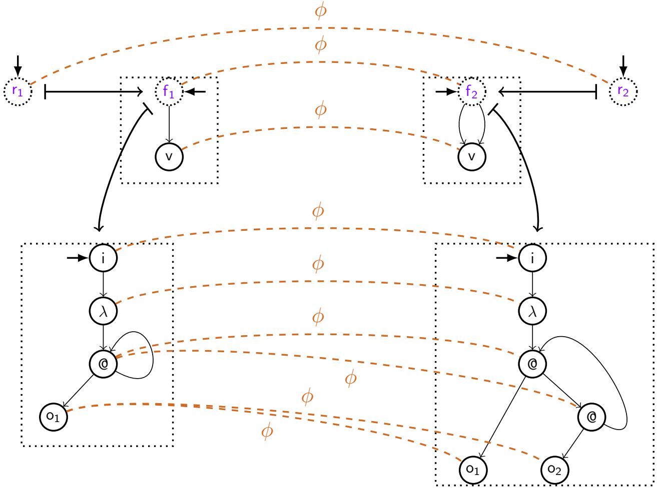
bisimulation (for intensional ntg-definition)



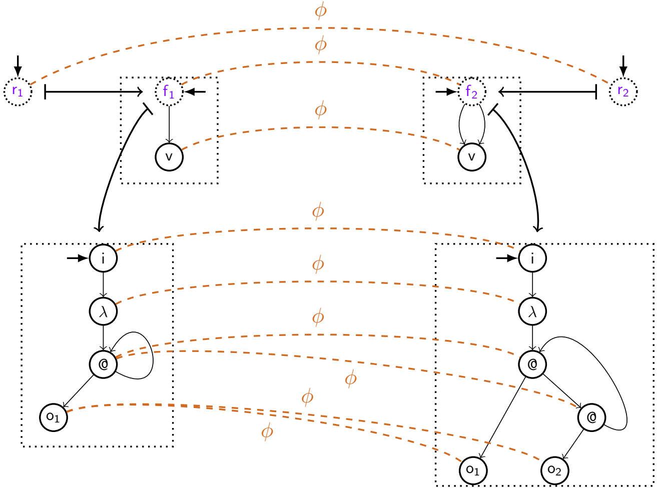
bisimulation (for intensional ntg-definition)



bisimulation (for intensional ntg-definition)



bisimulation (for intensional ntg-definition)



nested bisimulation, and rgs's versus ntgs

nested bisimilarity $\leftrightarrow^{\mathrm{ne}}$ on rgs's

- records nesting behaviour of rgs's via stacks of vertices
- easy: coincides with $\leftrightarrow$ on nested term graphs
- while conceptually finer, actually coincides with $\leftrightarrow$ also on rgs's

nested term graph $\mathcal{N}(\mathcal{R})$ induced by an rgs $\mathcal{R}$:

- obtained from the tree-unfolding of the dependency ARS by copying shared graph specifications

Theorem

Let Σ_1 and Σ_2 be ntg-signatures with same part Σ_{at} for atomic symbols.

For all rgs's $\mathcal{R}_1$ over Σ_1, and $\mathcal{R}_2$ over Σ_2, the following are equivalent:

(i) $\mathcal{R}_1 \leftrightarrow \mathcal{R}_2$;

(ii) $\mathcal{R}_1 \leftrightarrow^{\mathrm{ne}} \mathcal{R}_2$;

(iii) $\mathcal{N}(\mathcal{R}_1) \simeq \mathcal{N}(\mathcal{R}_2)$;

nested bisimulation, and rgs's versus ntgs

nested bisimilarity $\underline{\leftrightarrow}^{\mathsf{ne}}$ on rgs's

- records nesting behaviour of rgs's via stacks of vertices
- easy: coincides with $\underline{\leftrightarrow}$ on nested term graphs
- while conceptually finer, actually coincides with $\underline{\leftrightarrow}$ also on rgs's

nested term graph $\mathcal{N}(\mathcal{R})$ induced by an rgs $\mathcal{R}$:

- obtained from the tree-unfolding of the dependency ARS by copying shared graph specifications

Theorem

Let Σ_1 and Σ_2 be ntg-signatures with same part Σ_{at} for atomic symbols.

For all rgs's $\mathcal{R}_1$ over Σ_1, and $\mathcal{R}_2$ over Σ_2, the following are equivalent:

(i) $\mathcal{R}_1 \underline{\leftrightarrow} \mathcal{R}_2$;

(ii) $\mathcal{R}_1 \underline{\leftrightarrow}^{\mathsf{ne}} \mathcal{R}_2$;

(iii) $\mathcal{N}(\mathcal{R}_1) \simeq \mathcal{N}(\mathcal{R}_2)$;

nested bisimulation, and rgs's versus ntgs

nested bisimilarity $\underline{\leftrightarrow}^{\mathsf{ne}}$ on rgs's

- records nesting behaviour of rgs's via stacks of vertices
- easy: coincides with $\underline{\leftrightarrow}$ on nested term graphs
- while conceptually finer, actually coincides with $\underline{\leftrightarrow}$ also on rgs's

nested term graph $\mathcal{N}(\mathcal{R})$ induced by an rgs $\mathcal{R}$:

- obtained from the tree-unfolding of the dependency ARS by copying shared graph specifications

Theorem

Let Σ_1 and Σ_2 be ntg-signatures with same part Σ_{at} for atomic symbols.

For all rgs's $\mathcal{R}_1$ over Σ_1, and $\mathcal{R}_2$ over Σ_2, the following are equivalent:

(i) $\mathcal{R}_1 \underline{\leftrightarrow} \mathcal{R}_2$;

(ii) $\mathcal{R}_1 \underline{\leftrightarrow}^{\mathsf{ne}} \mathcal{R}_2$;

(iii) $\mathcal{N}(\mathcal{R}_1) \simeq \mathcal{N}(\mathcal{R}_2)$;

implementation by first-order term graphs

Theorem

Let Σ be an ntg-signature, and $\Sigma' = \Sigma \cup I \cup \{\mathsf{o}/2, \mathsf{i_r}/1, \mathsf{o_r}/1\}$.

There is a function $T : \mathrm{NG}(\Sigma) \to \mathrm{TG}(\Sigma')$ such that:

(i) *T preserves and reflects $\underline{\leftrightarrow}$.*

(ii) *T is efficiently computable.*

Proof based on the following definition of T on given nested term graph:

1. Pre-Processing: constant symbol vertices are linked to additional output vertex per nested vertex; continued outwards until top level;
2. Replacement/Adding Backlinks: starting on $rec(\mathrm{r})$, repeatedly replacing, a vertex v with a nested symbol f by the specification $rec(f)$ of f, thereby:
   - directing incoming edges at v to the root v_r of $rec(f)$
   - replacing output vertices o_i of $rec(f)$ by the binary symbol o with first edge to i-th successor of v, the second edge a back-link to v_r.

implementation by first-order term graphs

Theorem

Let Σ be an ntg-signature, and $\Sigma' = \Sigma \cup I \cup \{\mathrm{o}/2, \mathrm{i_r}/1, \mathrm{o_r}/1\}$.

There is a function $T : \mathrm{NG}(\Sigma) \to \mathrm{TG}(\Sigma')$ such that:

(i) *T preserves and reflects $\underline{\leftrightarrow}$.*

(ii) *T is efficiently computable.*

Proof based on the following definition of T on given nested term graph:

1. Pre-Processing: constant symbol vertices are linked to additional output vertex per nested vertex; continued outwards until top level;
2. Replacement/Adding Backlinks: starting on $rec(\mathrm{r})$, repeatedly replacing, a vertex v with a nested symbol f by the specification $rec(f)$ of f, thereby:
   - directing incoming edges at v to the root v_r of $rec(f)$
   - replacing output vertices o_i of $rec(f)$ by the binary symbol o with first edge to i-th successor of v, the second edge a back-link to v_r.

implementation by first-order term graphs

Theorem

Let Σ be an ntg-signature, and $\Sigma' = \Sigma \cup I \cup \{\mathrm{o}/2, \mathrm{i_r}/1, \mathrm{o_r}/1\}$.

There is a function $T : \mathrm{NG}(\Sigma) \to \mathrm{TG}(\Sigma')$ such that:

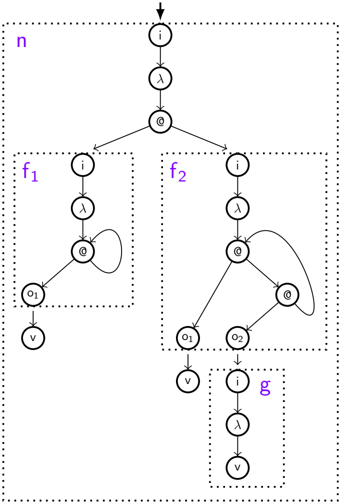
(i) *T preserves and reflects $\underline{\leftrightarrow}$.*

(ii) *T is efficiently computable.*

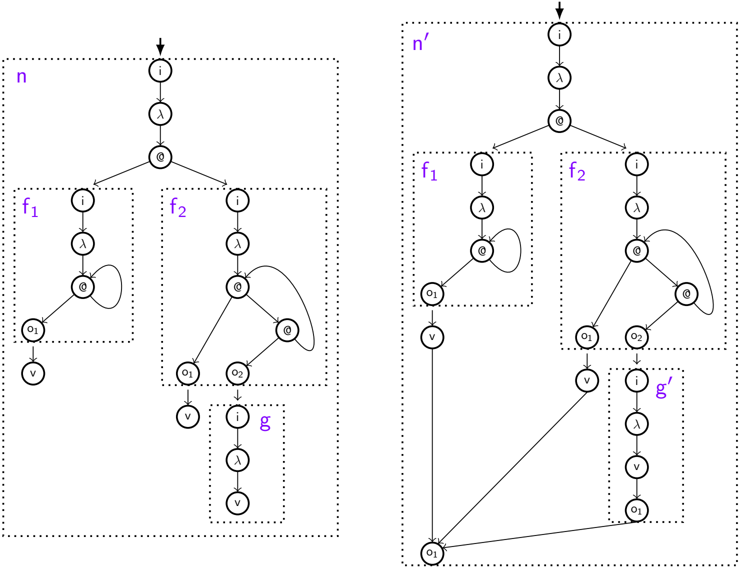
Proof based on the following definition of T on given nested term graph:

1. Pre-Processing: constant symbol vertices are linked to additional output vertex per nested vertex; continued outwards until top level;
2. Replacement/Adding Backlinks: starting on $rec(\mathrm{r})$, repeatedly replacing, a vertex v with a nested symbol f by the specification $rec(f)$ of f, thereby:
   - directing incoming edges at v to the root v_r of $rec(f)$
   - replacing output vertices o_i of $rec(f)$ by the binary symbol o with first edge to i-th successor of v, the second edge a back-link to v_r.

implementation by first-order term graph



implementation by first-order term graph



implementation by first-order term graphs

Theorem

Let Σ be an ntg-signature, and $\Sigma' = \Sigma \cup I \cup \{\mathrm{o}/2, \mathrm{i_r}/1, \mathrm{o_r}/1\}$.
There is a function $T : \mathrm{NG}(\Sigma) \to \mathrm{TG}(\Sigma')$ such that:

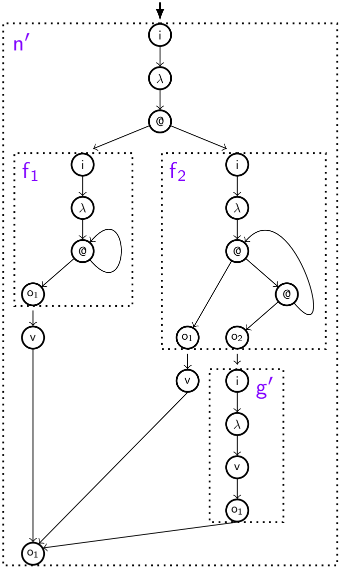
(i) *T preserves and reflects $\rightrightarrows$, and hence $\leftrightarrows$.*

(ii) *T is efficiently computable.*

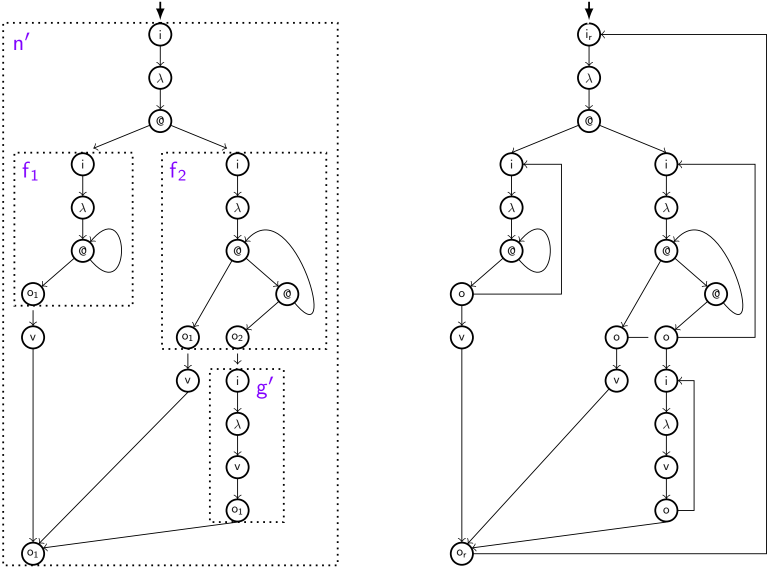
Proof based on the following definition of T on given nested term graph:

1. Pre-processing: constant symbol vertices are linked to additional output vertex per nested vertex; continued outwards until top level;
2. Replacement/Adding Backlinks: starting on $rec(\mathrm{r})$, repeatedly replacing, a vertex v with a nested symbol f by the specification $rec(f)$ of f, thereby:
   - directing incoming edges at v to the root v_r of $rec(f)$
   - replacing output vertices o_i of $rec(f)$ by the binary symbol o with first edge to i-th successor of v, the second edge a back-link to v_r.

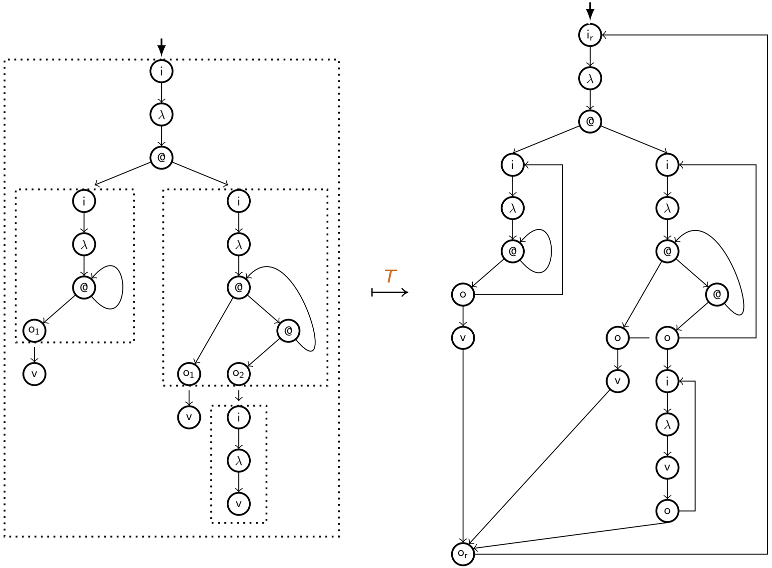
implementation by first-order term graph



implementation by first-order term graph



implementation by first-order term graph



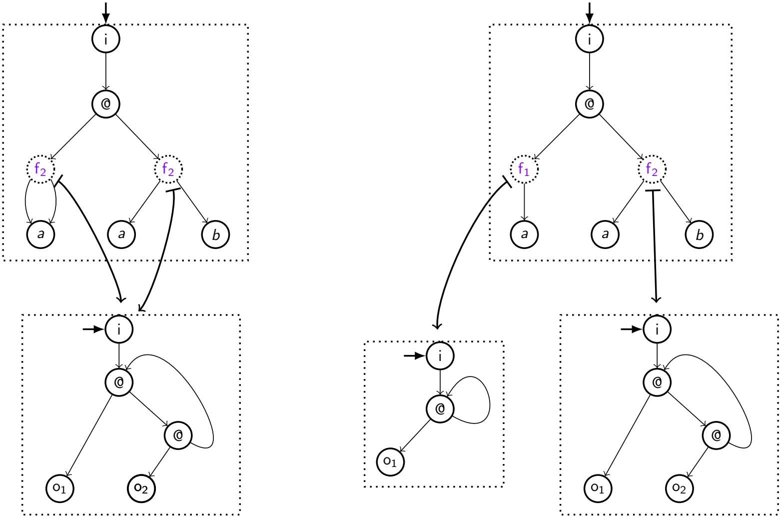
transfer of results from f-o term graphs

Corollary

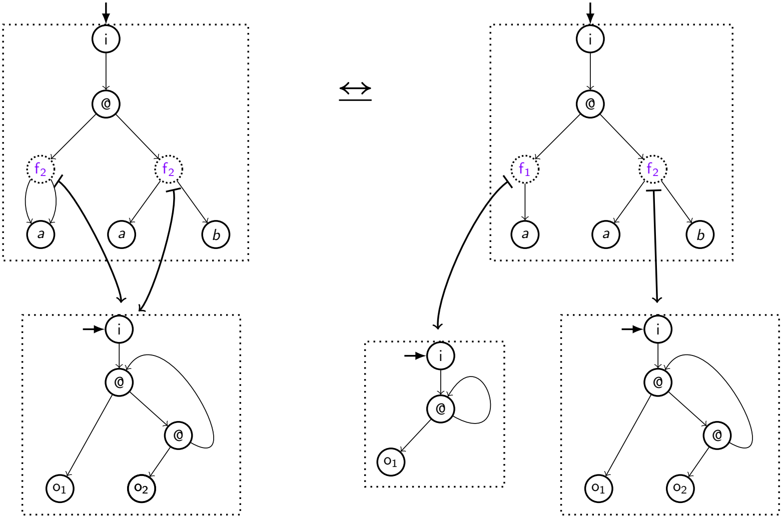
Let $\mathcal{N}$ be a nested term graph.

1. *$\mathcal{N}$ has, up to isomorphism, a unique nested term graph collapse.*
2. *The bisimulation equivalence class of $\mathcal{N}$ (up to isomorphism) forms a complete lattice w.r.t. $\underline{\rightarrow}$.*

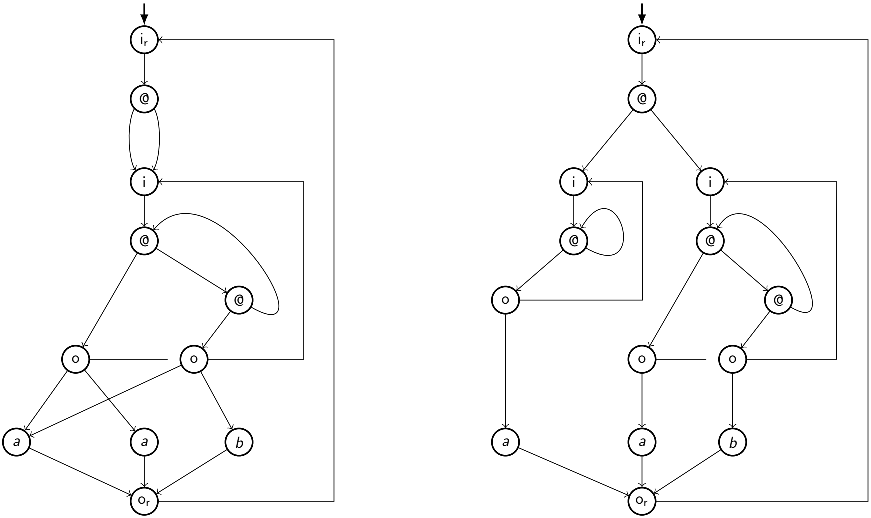
implementation fails for rgs's



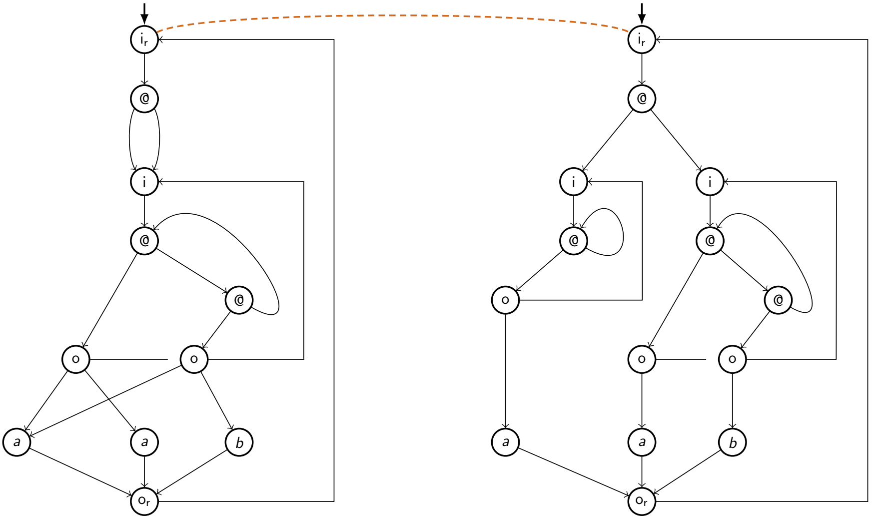
implementation fails for rgs's



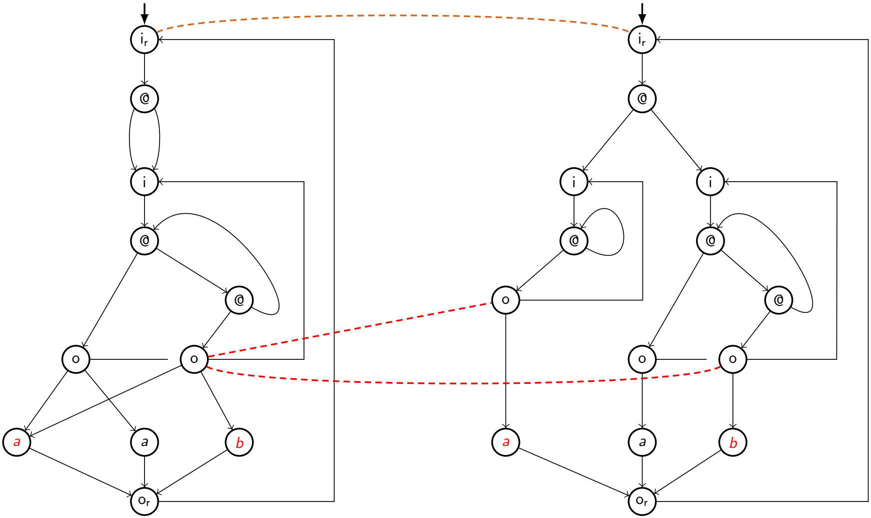
implementation fails for rgs's



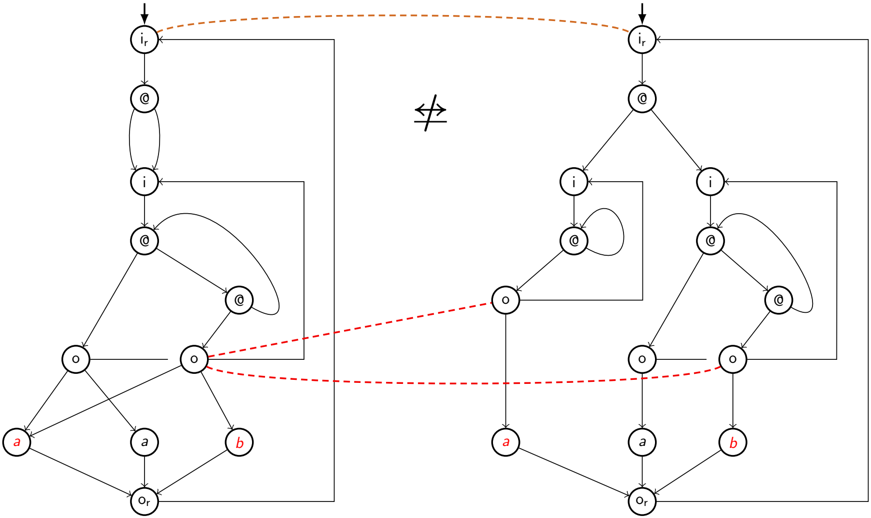
implementation fails for rgs's



implementation fails for rgs's



implementation fails for rgs's



further investigations and aims

- relation with similar concepts
  - proof nets and proof net reduction
- context-free graph grammars
  - view rgs's as context-free graph grammars
  - recognize rgs-generated nested term graphs as context-free graphs
- monadic formulation
  - nested term graphs as monads over some signature
  - categorically describe the implementation as first-order term graphs
- rewrite theory
  - higher-order terms interpreted as nested term graphs
  - implementation of h-o term rewriting as:
    - 'nested term graph rewriting'
    - then realization by f-o term graph (or port graph) rewriting
  - test-case: λ-calculus

further investigations and aims

- relation with similar concepts
  - proof nets and proof net reduction
- context-free graph grammars
  - view rgs's as context-free graph grammars
  - recognize rgs-generated nested term graphs as context-free graphs
- monadic formulation
  - nested term graphs as monads over some signature
  - categorically describe the implementation as first-order term graphs
- rewrite theory
  - higher-order terms interpreted as nested term graphs
  - implementation of h-o term rewriting as:
    - 'nested term graph rewriting'
    - then realization by f-o term graph (or port graph) rewriting
  - test-case: λ-calculus

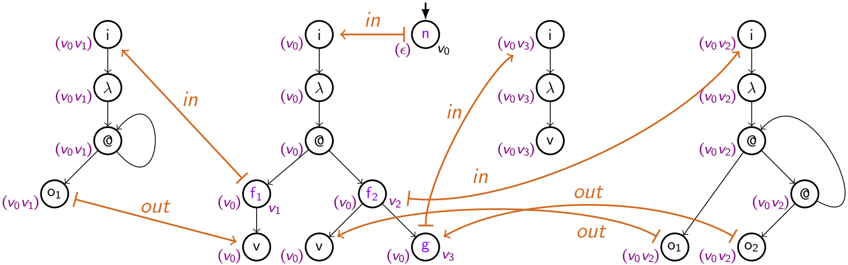
further investigations and aims

- relation with similar concepts
  - proof nets and proof net reduction
- context-free graph grammars
  - view rgs's as context-free graph grammars
  - recognize rgs-generated nested term graphs as context-free graphs
- monadic formulation
  - nested term graphs as monads over some signature
  - categorically describe the implementation as first-order term graphs
- rewrite theory
  - higher-order terms interpreted as nested term graphs
  - implementation of h-o term rewriting as:
    - 'nested term graph rewriting'
    - then realization by f-o term graph (or port graph) rewriting
  - test-case: λ-calculus

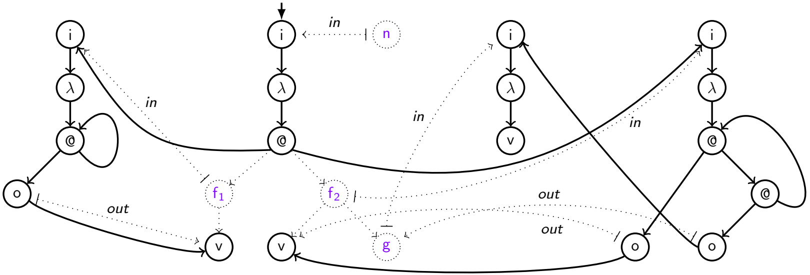
further investigations and aims

- relation with similar concepts
  - proof nets and proof net reduction
- context-free graph grammars
  - view rgs's as context-free graph grammars
  - recognize rgs-generated nested term graphs as context-free graphs
- monadic formulation
  - nested term graphs as monads over some signature
  - categorically describe the implementation as first-order term graphs
- rewrite theory
  - higher-order terms interpreted as nested term graphs
  - implementation of h-o term rewriting as:
    - 'nested term graph rewriting'
    - then realization by f-o term graph (or port graph) rewriting
  - test-case: λ-calculus

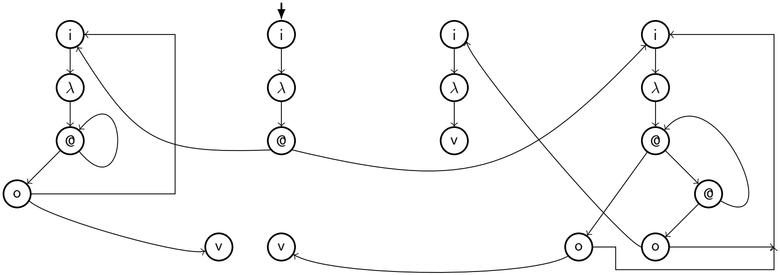
implementation by first-order term graph (via entg)



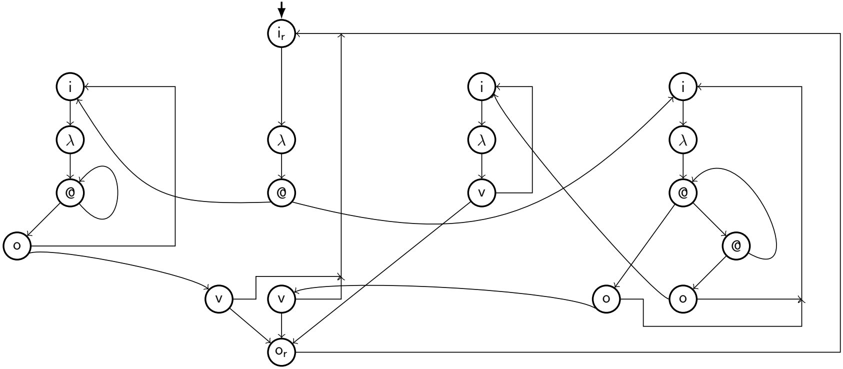
implementation by first-order term graph (via entg)



implementation by first-order term graph (via entg)



implementation by first-order term graph (via entg)



implementation by first-order term graph (via entg)