Answer Set Programming Graphs for Answer Set Programs Construction of Explanation Graphs from Extended Dependency Graphs Choosing Assumptions Answer Set Programming Graphs for Answer Set Programs Construction of Explanation Graphs from Extended Dependency Graphs Choosing Assumptions

Construction of Explanation Graphs from Extended Dependency Graphs for Answer Set Programs

Ella Albrecht, Patrick Krümpelmann, Gabriele Kern-Isberner

Lehrstuhl Informatik I, Technische Universität Dortmund

11. September 2013

Overview

Outline

Answer Set Programming

Definition (Extended Logic Program)

An *extended logic program* P is a set of rules r of the form
$r : h \leftarrow a_1, \ldots, a_m, not\ b_1, \ldots, not\ b_n$.

- $body(r) = \{a_1, \ldots, a_m, b_1, \ldots, b_n\}$
- $pos(r) = \{a_1, \ldots, a_m\}$, $neg(r) = \{b_1, \ldots, b_n\}$
- *herbrand basis* $\mathscr{H}(P)$: set of all grounded literals of P

Definition (Answer Set)

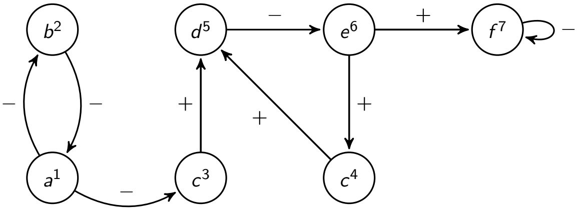
- $M \subseteq \mathscr{H}(P)$: consistent set of literals
- M is *closed* under P if $head(r) \in M$ whenever $pos(r) \subseteq M$ and $neg(r) \cap M = \emptyset$ for all rules $r \in P$
- M is an *answer set* of P if M is closed and minimal w.r.t. set inclusion

Outline

Extended Dependency Graphs

Definition (Extended Dependency Graph)

The EDG for a logic program P is a directed graph $EDG(P) = (V, E)$ with nodes V and edges $E \subseteq V \times V \times \{+, -\}$.



$r_1: \; a \leftarrow not\ b.$

$r_2: \; b \leftarrow not\ a.$

$r_3: \; c \leftarrow not\ a.$

$r_4: \; c \leftarrow e.$

$r_5: \; d \leftarrow c.$

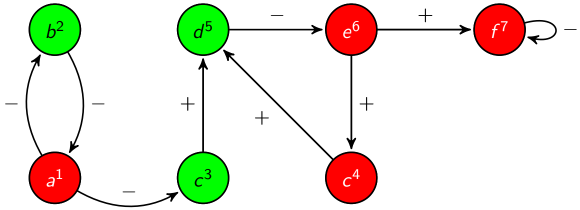
$r_6: \; e \leftarrow not\ d.$

$r_7: \; f \leftarrow e,\ not\ f.$

Extended Dependency Graphs

Definition (Extended Dependency Graph)

The EDG for a logic program P is a directed graph $EDG(P) = (V, E)$ with nodes V and edges $E \subseteq V \times V \times \{+, -\}$.



$r_1: \; a \leftarrow not\ b.$

$r_2: \; b \leftarrow not\ a.$

$r_3: \; c \leftarrow not\ a.$

$r_4: \; c \leftarrow e.$

$r_5: \; d \leftarrow c.$

$r_6: \; e \leftarrow not\ d.$

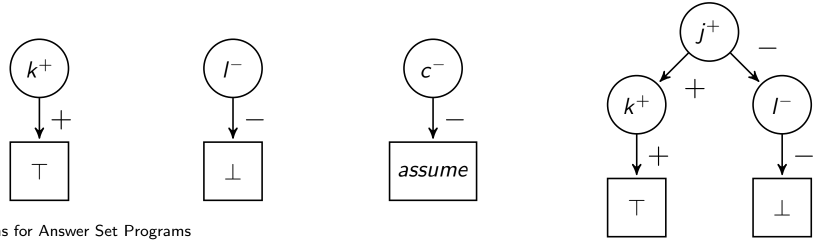
$r_7: \; f \leftarrow e,\ not\ f.$

Explanation Graph

Definition (Explanation Graph)

An *Explanation Graph* for a literal $a \in \mathcal{H}^p \cup \mathcal{H}^n$ in a program P w.r.t. an answer set M and an assumption $U \subseteq \mathcal{H}(P)$ is a directed graph $G = (V, E)$

- node set $V \subseteq \mathcal{H}^p \cup \mathcal{H}^n \cup \{\top, \bot, assume\}$
- edge set $E \subseteq V \times V \times \{+, -\}$
- $\mathcal{H}^p = \{a^+ \mid a \in \mathcal{H}(P)\}$, $\mathcal{H}^n = \{a^- \mid \mathcal{H}(P)\}$.

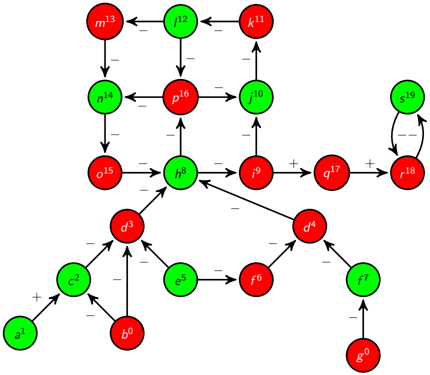


Outline

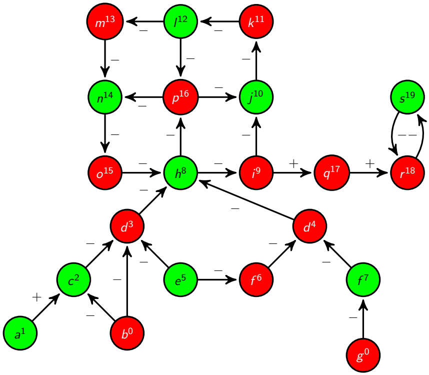
Transformation EDG ⇒ EG

1. Removing irrelevant edges and nodes
2. Marking nodes
3. Construct Explanation Graphs differing five cases of transformation
4. Determination of assumptions
5. Fitting graph to chosen assumption
6. Continue marking and construction process

Transformation EDG ⇒ EG - Removing irrelevant edges and nodes



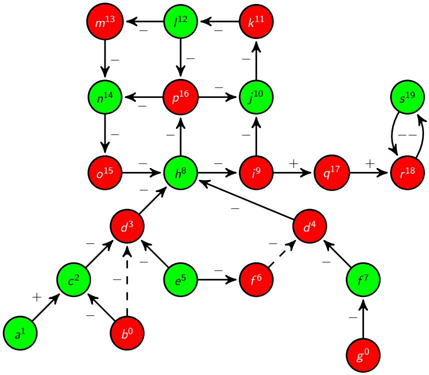
Transformation EDG ⇒ EG - Removing irrelevant edges and nodes



- **Irrelevant edge**: An edge (a_i^k, a_j^l, s) is *irrelevant* if

$v(a_i^k)$	$v(a_j^l)$	s
green	*green*	$-$
green	*red*	$+$
red	*green*	$+$
red	*red*	$-$

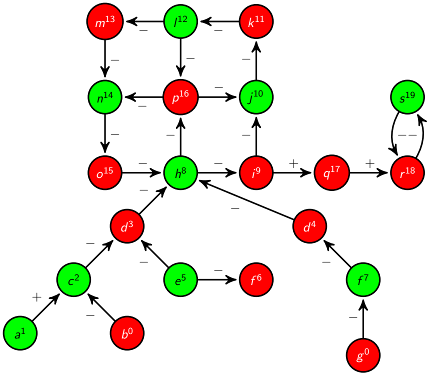
Transformation EDG ⇒ EG - Removing irrelevant edges and nodes



- **Irrelevant edge**: An edge (a_i^k, a_j^l, s) is *irrelevant* if

$\nu(a_i^k)$	$\nu(a_j^l)$	s
green	*green*	$-$
green	*red*	$+$
red	*green*	$+$
red	*red*	$-$

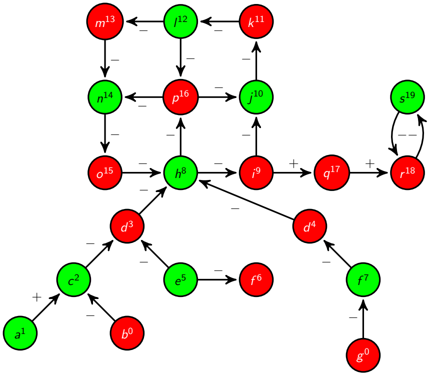
Transformation EDG ⇒ EG - Removing irrelevant edges and nodes



- **Irrelevant edge**: An edge (a_i^k, a_j^l, s) is *irrelevant* if

$v(a_i^k)$	$v(a_j^l)$	s
green	*green*	$-$
green	*red*	$+$
red	*green*	$+$
red	*red*	$-$

Transformation EDG ⇒ EG - Removing irrelevant edges and nodes

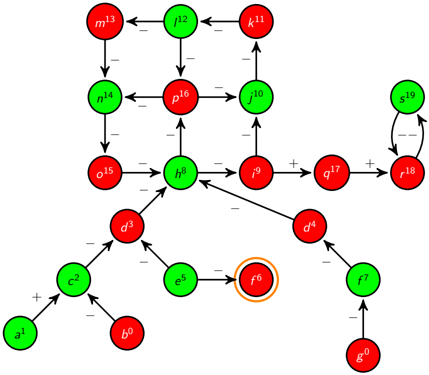


- **Irrelevant edge**: An edge (a_i^k, a_j^l, s) is *irrelevant* if

$\nu(a_i^k)$	$\nu(a_j^l)$	s
green	*green*	$-$
green	*red*	$+$
red	*green*	$+$
red	*red*	$-$

- **Irrelevant node**: A node a_i^k is *irrelevant* if $\nu(a_i^k) = red$ and there exists $l > 0$ where $\nu(a_i^l) = green$.

Transformation EDG ⇒ EG - Removing irrelevant edges and nodes

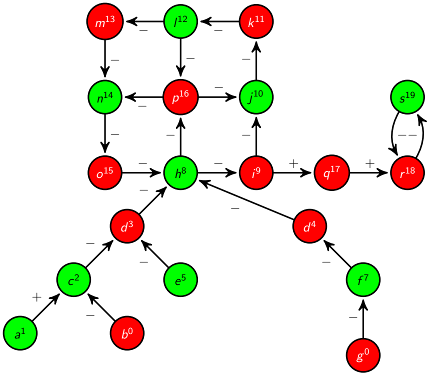


- **Irrelevant edge**: An edge (a_i^k, a_j^l, s) is *irrelevant* if

$\nu(a_i^k)$	$\nu(a_j^l)$	s
green	*green*	$-$
green	*red*	$+$
red	*green*	$+$
red	*red*	$-$

- **Irrelevant node**: A node a_i^k is *irrelevant* if $\nu(a_i^k) =$ *red* and there exists $l > 0$ where $\nu(a_i^l) =$ *green*.

Transformation EDG ⇒ EG - Removing irrelevant edges and nodes

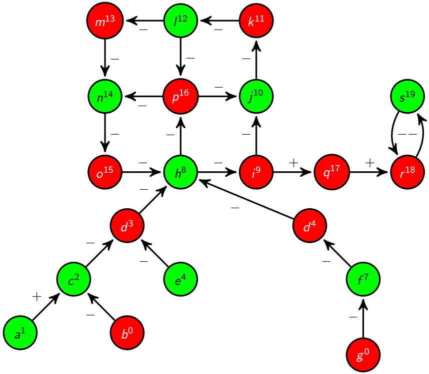


- **Irrelevant edge**: An edge (a_i^k, a_j^l, s) is *irrelevant* if

$\nu(a_i^k)$	$\nu(a_j^l)$	s
green	*green*	$-$
green	*red*	$+$
red	*green*	$+$
red	*red*	$-$

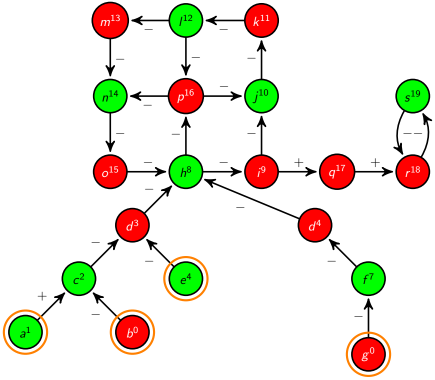
- **Irrelevant node**: A node a_i^k is *irrelevant* if $\nu(a_i^k) = red$ and there exists $l > 0$ where $\nu(a_i^l) = green$.

Transformation EDG ⇒ EG



- Start marking process at nodes without incoming edges

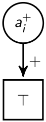
Transformation EDG ⇒ EG



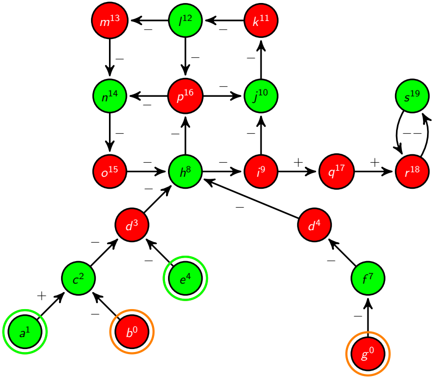
- Start marking process at nodes without incoming edges

Transformation EDG $\Rightarrow$ EG

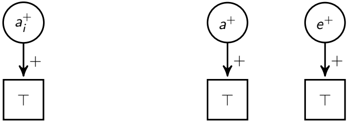
- Start marking process at nodes without incoming edges
- Transformation of fact nodes a_i^k



Transformation EDG ⇒ EG

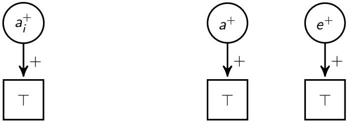


- Start marking process at nodes without incoming edges
- Transformation of fact nodes a_i^k

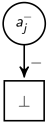


Transformation EDG $\Rightarrow$ EG

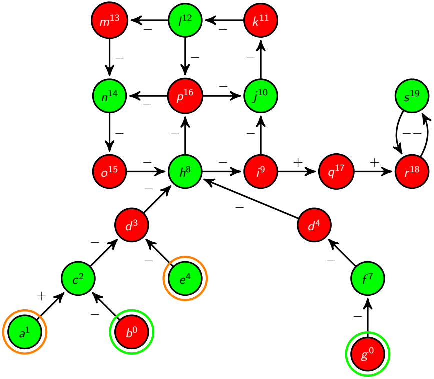
- Start marking process at nodes without incoming edges
- Transformation of fact nodes a_i^k



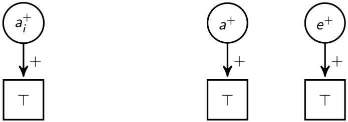
- Transformation of unfounded nodes a_j^l



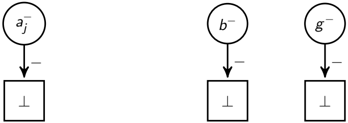
Transformation EDG ⇒ EG



- Start marking process at nodes without incoming edges
- Transformation of fact nodes a_i^k



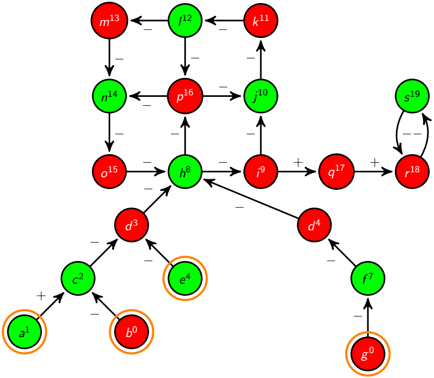
- Transformation of unfounded nodes a_j^l



Transformation EDG ⇒ EG

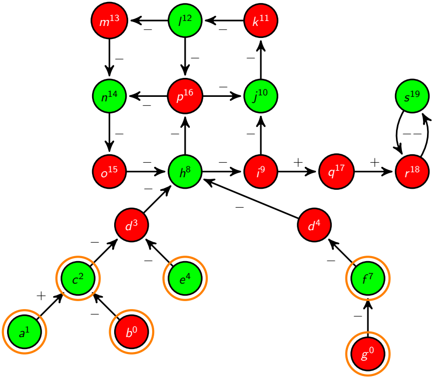
- A green node a_i^k can be marked if
  - all predecessor nodes a_j^l marked
  - if $\nu(a_j^l) = \mathit{green}$: there exists n, such that a_j^n marked

Transformation EDG $\Rightarrow$ EG



- A green node a_i^k can be marked if
  - all predecessor nodes a_j^l marked
  - if $\nu(a_j^l) = green$: there exists n, such that a_j^n marked

Transformation EDG ⇒ EG



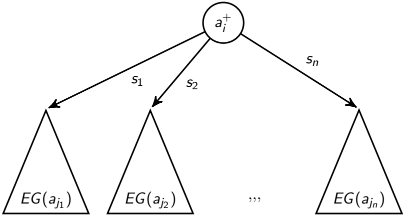
- A green node a_i^k can be marked if
  - all predecessor nodes a_j^l marked
  - if $\nu(a_j^l) = green$: there exists n, such that a_j^n marked

Transformation EDG ⇒ EG

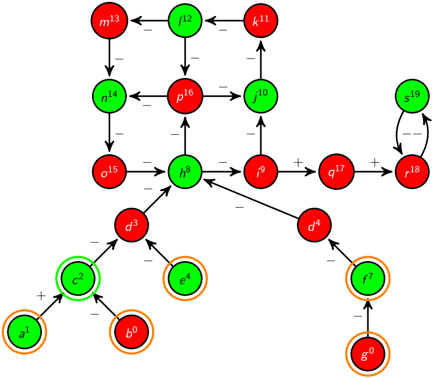
- Local consistent Explanation (LCE) to a literal a_i: minimal set of literals which directly influence truth value of a_i
- LCE for a green node: all direct predecessors

Transformation EDG ⇒ EG

- Local consistent Explanation (LCE) to a literal a_i: minimal set of literals which directly influence truth value of a_i
- LCE for a green node: all direct predecessors
- Transformation of dependent green nodes a_i^k

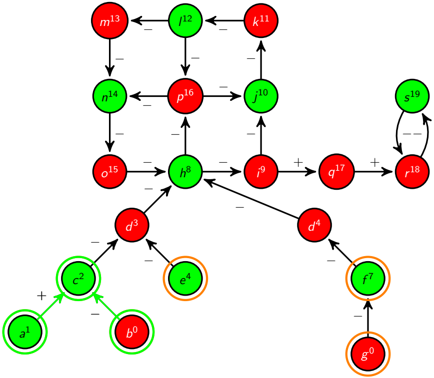


Transformation EDG ⇒ EG

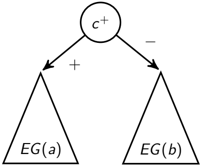


- Local consistent Explanation (LCE) to a literal a_i: minimal set of literals which directly influence truth value of a_i
- LCE for a green node: all direct predecessors
- Transformation of dependent green nodes a_i^k

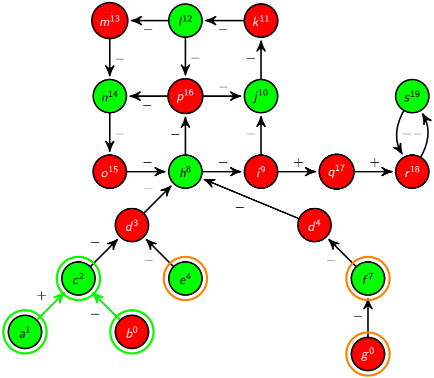
Transformation EDG ⇒ EG



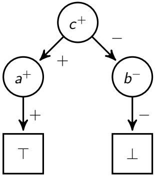
- Local consistent Explanation (LCE) to a literal a_i: minimal set of literals which directly influence truth value of a_i
- LCE for a green node: all direct predecessors
- Transformation of dependent green nodes a_i^k



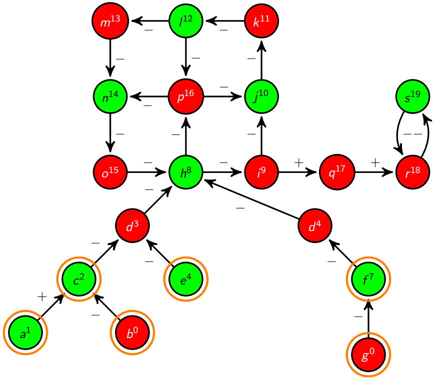
Transformation EDG ⇒ EG



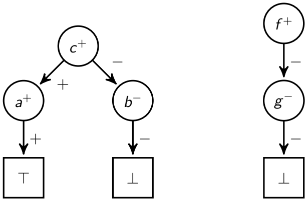
- Local consistent Explanation (LCE) to a literal a_i: minimal set of literals which directly influence truth value of a_i
- LCE for a green node: all direct predecessors
- Transformation of dependent green nodes a_i^k



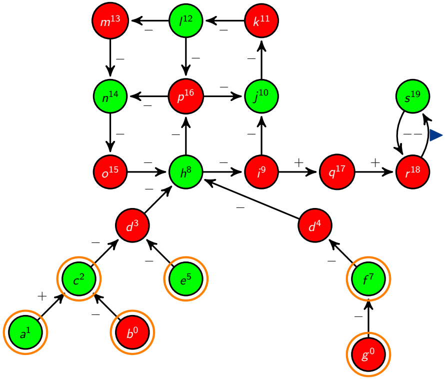
Transformation EDG ⇒ EG



- Local consistent Explanation (LCE) to a literal a_i: minimal set of literals which directly influence truth value of a_i
- LCE for a green node: all direct predecessors
- Transformation of dependent green nodes a_i^k

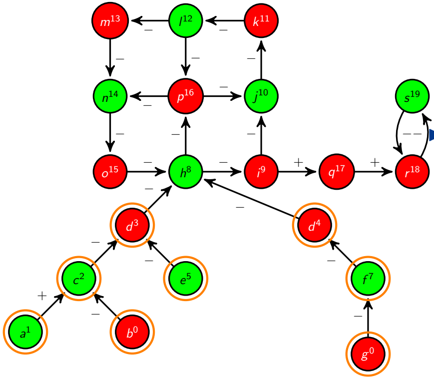


Transformation EDG ⇒ EG



- A red node a_i^k can be marked if
  - at least one predecessor node a_j^l marked
  - if $\nu(a_j^l) = \mathit{red}$: for all n node a_j^n marked

Transformation EDG ⇒ EG



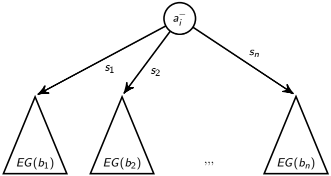
- A red node a_i^k can be marked if
  - at least one predecessor node a_j^l marked
  - if $v(a_j^l) = red$: for all n node a_j^n marked

Transformation EDG $\Rightarrow$ EG

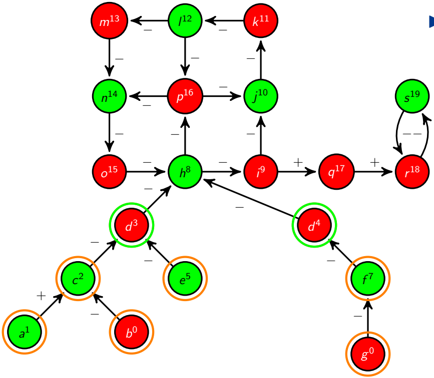
- A red node a_i^k can be marked if
  - at least one predecessor node a_j^l marked
  - if $v(a_j^l) = red$: for all n node a_j^n marked
- LCE for a red node a_i^k
  - considering all nodes representing a_i
  - one predecessor of each node

Transformation EDG ⇒ EG

- Transformation of dependent red nodes a_i^k



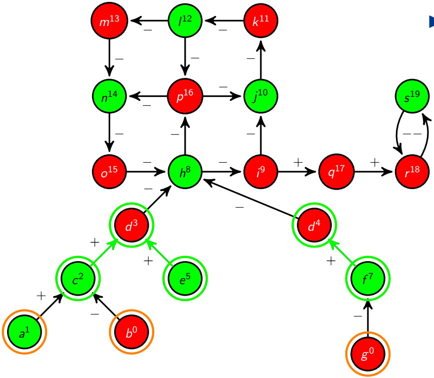
Transformation EDG ⇒ EG



- $$pn(d^3) = \{c^2, e^5\}$$
  $$pn(d^4) = \{f^7\}$$

$$L(d) = \{\{c, f\}, \{e, f\}\}$$

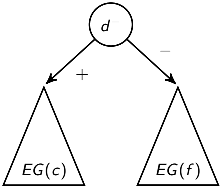
Transformation EDG ⇒ EG



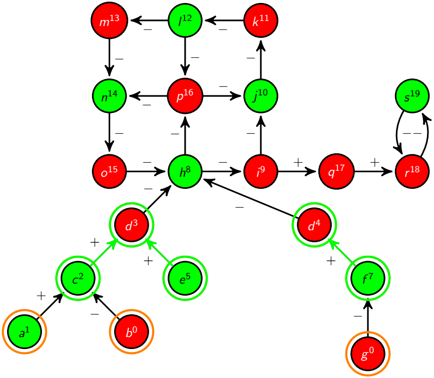
- $pn(d^3) = \{c^2, e^5\}$

  $pn(d^4) = \{f^7\}$

  $L(d) = \{\{c, f\}, \{e, f\}\}$



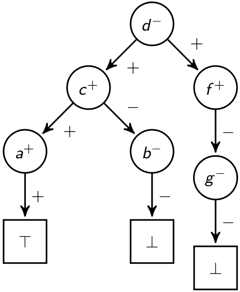
Transformation EDG ⇒ EG



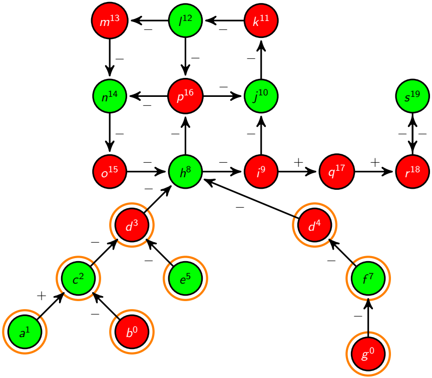
- $pn(d^3) = \{c^2, e^5\}$

  $pn(d^4) = \{f^7\}$

  $L(d) = \{\{c, f\}, \{e, f\}\}$



Well-founded model



Proposition

All unmarked nodes are undefined by the well-founded model.

Outline

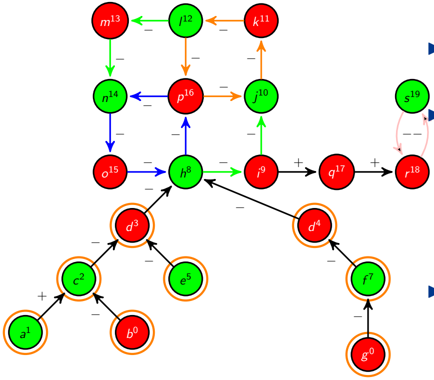
Assumptions

- **Tentative assumptions in an EDG**:
  $\mathcal{TA}(G') = \{a_i \mid a_i^k \in V', \nu(a_i^k) = red, (a_i^k, a_j^l, -) \in E, a_i^k \textit{ not marked}\}$
  - $G = (V, E)$: validly colored EDG
  - $G' = (V', E')$: EDG after removing irrelevant edges and nodes and marking nodes
- **Assumption**: Subset $U \subseteq \mathcal{TA}(G')$ such that all nodes in the EDG can be marked.
- Indeterminateness arises from cycles $\Rightarrow$ consideration of dependencies within and between cycles

Lemma

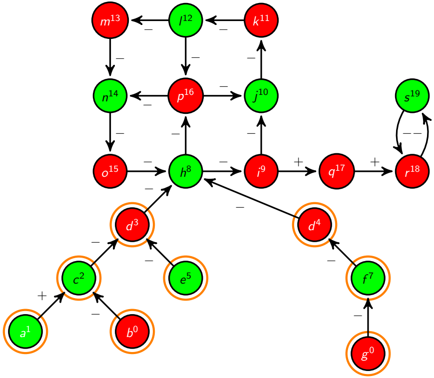
In an EDG without cycles all nodes can be marked.

Choosing Assumptions - Approach 1

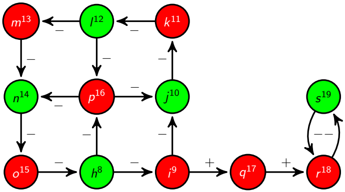


- One tentative assumption from each minimal cycle
- $C_1 = \{h^8, p^{16}, n^{14}, o^{15}\}$
  $C_2 = \{j^{10}, k^{11}, l^{12}, p^{16}\}$
  $C_3 = \{h^8, i^9, j^{10}, k^{11}, l^{12}, m^{13}, n^{14}, o^{15}\}$
  $C_4 = \{r^{18}, s^{19}\}$
- $Assumptions(G') =$ $\{\{p,k,r\},\{p,i,r\},\{p,m,r\},\{p,o,r\},\{o,k,r\}\}$

Choosing Assumptions - Approach 2



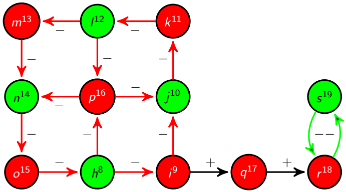
Choosing Assumptions - Approach 2



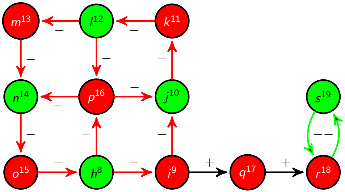
- **Linked cycles**: strongly connected components

Choosing Assumptions - Approach 2

- **Linked cycles**: strongly connected components

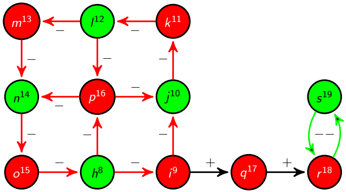


Choosing Assumptions - Approach 2



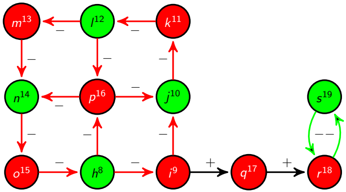
- **Linked cycles**: strongly connected components
- 1. Start at linked cycles without external incoming edges
  2. Find critical nodes
  3. Determine pre-conditions of each critical node
  4. Find successful combinations of pre-conditions
  5. Determine pre-condition paths and path assumptions
  6. Built assumptions for the linked cycle

Choosing Assumptions - Approach 2



- Critical nodes: $\{j^{10}, n^{14}\}$
- Successfull combination: $\{i^9, p^{16}\}$, $\{m^{13}, p^{16}\}$
- $$\mathscr{PA}(path(i^9)) = \{i, o\}$$
  $$\mathscr{PA}(path(p^{16})) = \{p, k, o\}$$
  $$\mathscr{PA}(path(m^{13})) = \{m, k\}$$
- $Assumptions(LC_{red}) =$
  $\{\{i,p\},\{i,k\},\{i,o\},\{o,p\},\{o,k\},\{o\}\} \cup$
  $\{\{p,m\},\{p,k\},\{k,m\},\{k\},\{o,m\},\{o,k\}\}$
- $\mu Assumptions(LC_{red}) = \{\{o\},\{k\},\{i,p\},\{p,m\}\}$

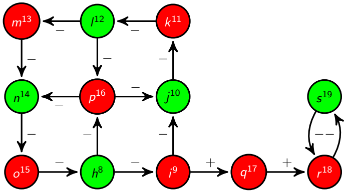
Choosing Assumptions - Approach 2



- Critical nodes: $\{j^{10}, n^{14}\}$
- Successfull combination: $\{i^9, p^{16}\}$, $\{m^{13}, p^{16}\}$
- $$\mathscr{PA}(path(i^9)) = \{i, o\}$$
  $$\mathscr{PA}(path(p^{16})) = \{p, k, o\}$$
  $$\mathscr{PA}(path(m^{13})) = \{m, k\}$$
- $Assumptions(LC_{red}) =$
  $\{\{i,p\},\{i,k\},\{i,o\},\{o,p\},\{o,k\},\{o\}\} \cup$
  $\{\{p,m\},\{p,k\},\{k,m\},\{k\},\{o,m\},\{o,k\}\}$
- $\mu Assumptions(LC_{red}) = \{\{o\},\{k\},\{i,p\},\{p,m\}\}$

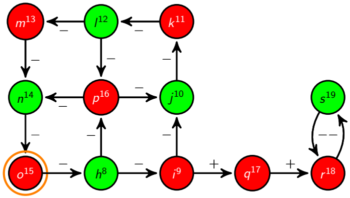
Choosing Assumptions - Approach 2

- Assumption: o



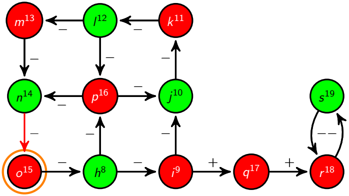
Choosing Assumptions - Approach 2

- Assumption: *o*



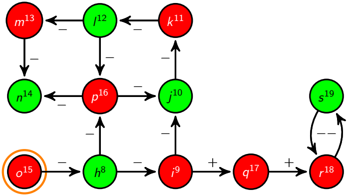
Choosing Assumptions - Approach 2

- Assumption: *o*



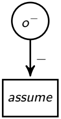
Choosing Assumptions - Approach 2

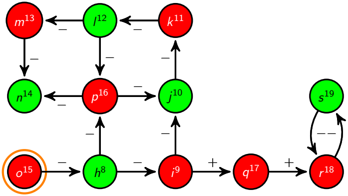
- Assumption: o



Choosing Assumptions - Approach 2

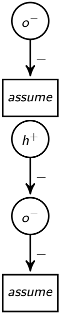
- Assumption: o

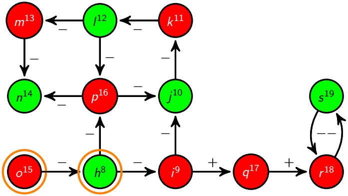




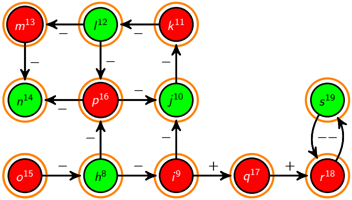
Choosing Assumptions - Approach 2

- Assumption: *o*

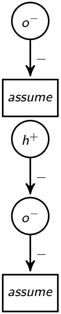




Choosing Assumptions - Approach 2



- Assumption: o



- Two cases:
  1. All nodes of the EDG marked ⇒ DONE
  2. There are still unmarked nodes in the EDG ⇒ Remove marked nodes and start again with the determination of assumptions

Summary

- Construction of Explanation Graphs from validly colored Extended Dependency Graphs
  - Marking nodes
  - Five types of transformation
- Choosing assumptions
  - Approach 1: linear, but non-minimal
  - Approach 2: exponential, but minimal