Mdag-Based Network Reliability Evaluation

Jacek Jonczy

Reasoning under UNcertainty Group
Institute of Computer Science and Applied Mathematics
University of Berne, Switzerland
http://www.iam.unibe.ch/~run/index.html
email: jonczy@iam.unibe.ch

Seminar TCS - RUN
Fribourg

October 3rd, 2007
Outline

1. Introduction
2. Structure Function Representation
3. Network Reliability Evaluation
4. Multilayer Network Model
5. Conclusion
Outline

1. Introduction
2. Structure Function Representation
3. Network Reliability Evaluation
4. Multilayer Network Model
5. Conclusion
Network Model: probabilistic graph $G = (V, E)$

Following assumptions:
- Networks are directed
- Nodes are perfectly reliable
- Edges fail independently with known probabilities

State space $\Omega = \{0, 1\}^m$

Network state $S = (s_1, \ldots, s_m) \in \Omega$ s.t.

$$s_i = \begin{cases} 
1 & \text{if edge } e_i \text{ is operational,} \\
0 & \text{otherwise.} 
\end{cases}$$

Associated Boolean function $\Phi : \Omega \rightarrow \{0, 1\}$

$S$ is operational (failure) state iff $\Phi(S) = 1$ (0)
structure function (SF)

One-to-one relation: system $\leftrightarrow$ structure function

Many ways to represent the structure function:
- pathset representation (DNF)
- dual view: cutset representation (CNF)
- pivotal decomposition (OBDD)
- ...

Goal: compact and efficient representation

Our approach: PDAG (MDAG) representation
Network Reliability Problems

- $Conn_2$: source-to-terminal connectedness
- $Conn_A$: source-to-all-terminal connectedness (reachability)
- $Conn_{\forall k}$: source-to-k-terminal connectedness
- $Conn_{\exists k}$: source-to-any-terminal connectedness
Network Reliability Problems

- \( Conn_2 \): source-to-terminal connectedness
- \( Conn_A \): source-to-all-terminal connectedness (reachability)
- \( Conn_{\forall k} \): source-to-k-terminal connectedness
- \( Conn_{\exists k} \): source-to-any-terminal connectedness
Network Reliability Problems

- \( Conn_2 \): source-to-terminal connectedness
- \( Conn_A \): source-to-all-terminal connectedness (reachability)
- \( Conn_{\forall k} \): source-to-k-terminal connectedness
- \( Conn_{\exists k} \): source-to-any-terminal connectedness
Network Reliability Problems

- $Conn_2$: source-to-terminal connectedness
- $Conn_A$: source-to-all-terminal connectedness (reachability)
- $Conn_{\forall k}$: source-to-k-terminal connectedness
- $Conn_{\exists k}$: source-to-any-terminal connectedness
Network Reliability Problems

- $Conn_2$: source-to-terminal connectedness
- $Conn_A$: source-to-all-terminal connectedness (reachability)
- $Conn_{\forall k}$: source-to-k-terminal connectedness
- $Conn_{\exists k}$: source-to-any-terminal connectedness
Existing Network Reliability Methods

- Enumeration methods
  - complete state, minpath and mincut enumeration, ...

- Transformation methods
  - Shannon’s decomposition, reductions, ...

- Direct methods
  - involving OBDD, EED, ...

- Approximation methods
  - estimation (Monte Carlo), bounding, ...
Outlook

1. Representation of a generic structure function
2. Polynomial time algorithm for generic SF
3. Reliability evaluation with PDAG-based SF
4. Multilayer network extension
Outline

1. Introduction
2. Structure Function Representation
3. Network Reliability Evaluation
4. Multilayer Network Model
5. Conclusion
Propositional DAGs (PDAGs)

- Graph-based language for the representation of Boolean Functions
- Example: PDAG $\varphi$ representing the odd-parity function $f_\varphi$ with respect to $V = \{a, b, c, d\}$

\[
f_\varphi = (a \oplus b) \oplus (c \oplus d)
\]

- $\text{PDAG}_V$: set of all possible PDAGs w.r.t. set of propositional variables $V$
PDAG Properties

- Decomposability (c) and determinism (d)
  - PDAGs satisfying the cd-properties constitute the sub-language cd-PDAG
- Succinctness (compactness)
  \[
  \text{PDAG} \prec \text{cd-PDAG} \left\{ \prec \text{FBDD} \prec \text{OBDD} \prec \text{d-DNF} \right\}
  \]
- Computing probabilities
  - \( \varphi \in \text{cd-PDAG} \) allows efficient (linear with respect to \( |\varphi| \)) probability computation \( p(\varphi) \)
PDAG Properties

- Decomposability (c) and determinism (d)
  - PDAGs satisfying the cd-properties constitute the sub-language cd-PDAG

- Succinctness (compactness)

  \[
  \text{PDAG} \prec \text{cd-PDAG} \prec \begin{cases} 
  \text{FBDD} \prec \text{OBDD} \\
  \prec \text{d-DNF}
  \end{cases}
  \]

- Computing probabilities
  - \( \varphi \in \text{cd-PDAG} \) allows efficient (linear with respect to \(|\varphi|\)) probability computation \( p(\varphi) \)
Outline

1. Introduction
2. Structure Function Representation
3. Network Reliability Evaluation
4. Multilayer Network Model
5. Conclusion
Overview

- Network Representation \( \mathcal{N} \)
- Phase 1
  - Generic Structure Function \( \varphi_s \)
- Phase 2
  - Specific Structure Function \( \varphi_{s,t} (\varphi_s, \forall t \varphi_s, \exists t) \)
- Phase 3a
  - Exact Computation
  - Step 1
    - \( \varphi_{s,t} \rightarrow \varphi_{s,t}^d \rightarrow \varphi_{s,t}^{cd} \)
  - Step 2
    - \( p(\varphi_{s,t}^{cd}) \)
- Phase 3b
  - Approximate Computation
Overview: Phase 1

Network Representation $\mathcal{N}$

Phase 1

Generic Structure Function $\varphi_s$

Phase 2

Specific Structure Function $\varphi_{s,t}$ ($\varphi_s \forall t \varphi_s \exists t$)

Phase 3a

Exact Computation

Step 1

$\varphi_{s,t} \rightarrow \varphi_{s,t}^d \rightarrow \varphi_{s,t}^{cd}$

Step 2

$P(\varphi_{s,t}^{cd})$

Phase 3b

Approximate Computation
Algorithm is essentially a variable (node) elimination process

Fix source node $A$, eliminate remaining nodes in order $D, C, B$

Order is chosen according to some heuristic

Algorithm **input**: reachability matrix $N$

\[
\begin{array}{c|cccc}
 & A & B & C & D \\
 A & \lambda_A & e_1 & e_2 & \\
 B & \bot & \lambda_B & \bot & e_3 \\
 C & \bot & e_5 & \lambda_C & e_4 \\
 D & \bot & \bot & \bot & \lambda_D \\
\end{array}
\]
Phase 1: Generating the Generic Structure Function

- Algorithm is essentially a variable (node) elimination process
- Fix source node $A$, eliminate remaining nodes in order $D, C, B$
- Order is chosen according to some heuristic
- Algorithm input: reachability matrix $\mathcal{N}$

![Diagram of a network with nodes A, B, C, D and edges e1, e2, e3, e4, e5.]

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$\lambda_A$</td>
<td>$e_1$</td>
<td>$e_2$</td>
<td>$\perp$</td>
</tr>
<tr>
<td>$B$</td>
<td>$\perp$</td>
<td>$\lambda_B$</td>
<td>$\perp$</td>
<td>$e_3$</td>
</tr>
<tr>
<td>$C$</td>
<td>$\perp$</td>
<td>$e_5$</td>
<td>$\lambda_C$</td>
<td>$e_4$</td>
</tr>
<tr>
<td>$D$</td>
<td>$\perp$</td>
<td>$\perp$</td>
<td>$\perp$</td>
<td>$\lambda_D$</td>
</tr>
</tbody>
</table>

Terminal selectors
Phase 1:
Generating the Generic Structure Function

- Algorithm is essentially a variable (node) elimination process
- Fix source node $A$, eliminate remaining nodes in order $D, C, B$
- Order is chosen according to some heuristic
- Eliminating $D$:

\[ A \xrightarrow{\lambda_A} B \]  
\[ B \xrightarrow{\lambda_B} C \]  
\[ C \xrightarrow{\lambda_C} D \]  
\[ D \xrightarrow{\lambda_D} \]

\[
\begin{array}{cccc}
A & B & C & D \\
A & \lambda_A & e_1 & e_2 & \perp \\
B & \perp & \lambda_B & \perp & e_3 \\
C & \perp & e_5 & \lambda_C & e_4 \\
D & \perp & \perp & \perp & \lambda_D \\
\end{array}
\]
Phase 1: Generating the Generic Structure Function

- Algorithm is essentially a variable (node) elimination process
- Fix source node $A$, eliminate remaining nodes in order $D, C, B$
- Order is chosen according to some heuristic
- After eliminating $D$:

$$
\begin{array}{cccc}
A & B & C & \cdots \\
\lambda_A & e_1 & e_2 & \cdots \\
\bot & \lambda_B \lor (e_3 \land \lambda_D) & \bot & \cdots \\
\bot & e_5 & \lambda_C \lor (e_4 \land \lambda_D) & \cdots \\
\vdots & \vdots & \vdots & \ddots
\end{array}
$$
Phase 1:
Generating the Generic Structure Function

- Algorithm is essentially a variable (node) elimination process
- Fix source node $A$, eliminate remaining nodes in order $D$, $C$, $B$
- Order is chosen according to some heuristic
- Eliminating $C$:
Phase 1: Generating the Generic Structure Function

- Algorithm is essentially a variable (node) elimination process
- Fix source node $A$, eliminate remaining nodes in order $D, C, B$
- Order is chosen according to some heuristic
- After eliminating $C$:

$$
\begin{array}{c}
A & \lambda_A \lor [e_2 \land (\lambda_C \lor (e_4 \land \lambda_D))] & e_1 \lor (e_2 \land e_5) \\
B & \bot & \lambda_B \lor (e_3 \land \lambda_D) \\
\vdots & \vdots & \vdots
\end{array}
$$

![Diagram of network with nodes A, B, C, D and edges e1, e2, e3, e4, e5.]
Phase 1:
Generating the Generic Structure Function

- Algorithm is essentially a variable (node) elimination process
- Fix source node $A$, eliminate remaining nodes in order $D, C, B$
- Order is chosen according to some heuristic
- Eliminating $B$:
Phase 1:
Generating the Generic Structure Function

- Algorithm is essentially a variable (node) elimination process
- Fix source node $A$, eliminate remaining nodes in order $D, C, B$
- Order is chosen according to some heuristic
- After eliminating $B$:

\[
A 
\begin{array}{c|c|c|c}
\lambda_A & \left[ e_2 \land (\lambda_C \lor (e_4 \land \lambda_D)) \right] \lor \left[ (e_1 \lor (e_2 \land e_5)) \land (\lambda_B \lor (e_3 \land \lambda_D)) \right]
\end{array}
\]

\[
A \quad \vdots
\]

\[
 \vdots
\]

\[
 \vdots
\]

\[
 \vdots
\]

\[
 \vdots
\]

\[
 \vdots
\]

\[
 \vdots
\]
Phase 1: Generating the Generic Structure Function

- Algorithm is essentially a variable (node) elimination process
- Fix source node $A$, eliminate remaining nodes in order $D$, $C$, $B$
- Order is chosen according to some heuristic
- Algorithm output: generic SF, PDAG representation:

```
ϕ_A
λ_A

λ_C

λ_D

λ_B
e_4

e_3
e_2
e_5
e_4
e_2
```
Overview: Phase 2

Network Representation $\mathcal{N}$

Phase 1

Generic Structure Function $\varphi_s$

Phase 2

Specific Structure Function $\varphi_{s,t}$ ($\varphi_s, \forall t \varphi_s, \exists t$)

Phase 3a

Exact Computation

Step 1

$\varphi_{s,t} \rightarrow \varphi_{s,t}^{d} \rightarrow \varphi_{s,t}^{cd}$

Step 2

$p(\varphi_{s,t}^{cd})$

Phase 3b

Approximate Computation

Step 1

$p_1(\varphi_{s,t})$
Phase 2: Generating Specific Structure Functions

- Starting point: generic SF with source node $A$, $\varphi_A$
- (1) let’s choose $D$ as terminal node
Phase 2: Generating Specific Structure Functions

- Starting point: generic SF with source node \( A, \varphi_A \)
- (2) instantiate \( \lambda_A \) to \( \lambda_D \) within \( \varphi_A \):

\[
\varphi_A \equiv \lambda_A \equiv \lambda_C \equiv \lambda_D \equiv \top \equiv \bot \equiv \bot \equiv \bot \equiv \bot
\]
Phase 2: Generating Specific Structure Functions

- Starting point: generic SF with source node $A$, $\varphi_A$
- (3) obtain SF $\varphi_{A,D}$ for $Conn_{A,D}$:
Overview: Phase 3a

Network Representation \( \mathcal{N} \)

Phase 1

Generic Structure Function \( \varphi_s \)

Phase 2

Specific Structure Function \( \varphi_{s,t} \ (\varphi_s \forall t \ \varphi_s \exists t) \)

Phase 3a

Step 1

\( \varphi_{s,t} \rightarrow \varphi_s \rightarrow \varphi_{s,t} \rightarrow \varphi_{s,t}^d \)

Step 2

\( p(\varphi_{s,t}^d) \)

Phase 3b

Approximate Computation
Phase 3a: Exact Reliability Computation

**Step 1:** $\varphi_{s,t} \xrightarrow{(a)} \varphi_{d}^{s,t} \xrightarrow{(b)} \varphi_{c,d}^{s,t}$ transformation
Phase 3a: Exact Reliability Computation

- **Step 1:** \( \phi_{s,t} \overset{(a)}{\rightarrow} \phi_{s,t}^{d} \overset{(b)}{\rightarrow} \phi_{s,t}^{cd} \) transformation
  - (a) make \( \phi_{A,D} \) deterministic

---

Diagram with labeled edges and nodes.
Phase 3a: Exact Reliability Computation

- **Step 1:** \( \varphi_{s,t} \xrightarrow{(a)} \varphi_{s,t}^d \xrightarrow{(b)} \varphi_{s,t}^{cd} \) transformation

  - (a) make \( \varphi_{A,D} \) **deterministic**: obtain \( \varphi_{A,D}^d \) (efficient)
Phase 3a: Exact Reliability Computation

- **Step 1:** $\varphi_{s,t} \xrightarrow{(a)} \varphi_{s,t}^d \xrightarrow{(b)} \varphi_{s,t}^{cd}$ transformation
  - (b) make $\varphi_{A,D}^d$ decomposable
Phase 3a:
Exact Reliability Computation

**Step 1:** \( \varphi_{s,t} \xrightarrow{(a)} \varphi_{s,t}^d \xrightarrow{(b)} \varphi_{s,t}^{cd} \) transformation

- (b) make \( \varphi_{A,D}^d \) decomposable: obtain \( \varphi_{A,D}^{cd} \) (hard in general)
Phase 3a, cont.
Exact Reliability Computation

- **Step 2**: compute reliability $p(\varphi^{cd}_{A,D})$
Step 2: compute reliability $p(\varphi_{A,D}^{cd})$

- assign probabilities to leaf-nodes
Phase 3a, cont.
Exact Reliability Computation

- **Step 2**: compute reliability $p(\varphi_{A,D}^{cd})$
  - compute recursively up to the root
Phase 3a, cont.

Exact Reliability Computation

- **Step 2:** compute reliability $p(\phi_{A,D}^c)$
  - compute recursively up to the root
**Phase 3a, cont.**

Exact Reliability Computation

- **Step 2:** compute reliability $p(\varphi_{A,D}^{cd})$

  - final result $Conn_{A,D}$ is 0.8147
Overview: Phase 3b

- **Network Representation** $\mathcal{N}$
- **Phase 1**
  - Generic Structure Function $\varphi_s$
- **Phase 2**
  - Specific Structure Function $\varphi_{s,t}$ ($\varphi_{s,\forall t} \quad \varphi_{s,\exists t}$)
- **Phase 3a**
  - Exact Computation
- **Phase 3b**
  - Approximate Computation

**Step 1**
- $\varphi_{s,t} \rightarrow \varphi_{s,t}^{d} \rightarrow \varphi_{s,t}^{cd}$

**Step 2**
- $p(\varphi_{s,t}^{cd})$
Phase 3b: Approximate Reliability Computation

- Based on SF representation $\varphi_{A,D}$
- No further transformations necessary
- Monte Carlo method:
  - Given network state $S \in \Omega$, generate $N$ random samples $S_1, \ldots, S_N$ according to $p(S)$
  - Estimate reliability by
    $$\frac{1}{N} \cdot \sum_{i=1}^{N} \Phi(S_i),$$
    where $\Phi$ is the structure function
Outline

1. Introduction
2. Structure Function Representation
3. Network Reliability Evaluation
4. Multilayer Network Model
5. Conclusion
Motivation: Railway Network

(1) Railway system:

(2) Electric power system:

- Consider a third layer: train connections
- Question: will a train be able to pass from $G$ to $K$?
Motivation: Railway Network

(1) Railway system:

(2) Electric power system:

- Consider a third layer: **train connections**
- Question: will a train be able to pass from $G$ to $K$?
Motivation: Railway Network

(1) Railway system:

(2) Electric power system:

- Consider a third layer: train connections
- Question: will a train be able to pass from $G$ to $K$?
Motivation: Internet Connection

(1) Physical layer:

(2) Internet connection:

- Asymmetric dependence: (1) is required for (2)
- Question: will $R_1$ be able to communicate with $R_2$?
Motivation: Internet Connection

(1) Physical layer:

(2) Internet connection:

- Asymmetric dependence: (1) is required for (2)
- Question: will $R_1$ be able to communicate with $R_2$?
Motivation: Internet Connection

(1) Physical layer:

(2) Internet connection:

- Asymmetric dependence: (1) is required for (2)
- Question: will $R_1$ be able to communicate with $R_2$?
Motivation: Credential Network

(1) Certificate Layer:

(2) Trust Layer:

Both layers are mutually dependent

Question: Is Bob’s public key authentic for Alice?
Motivation: Credential Network

(1) Certificate Layer:

(2) Trust Layer:

Both layers are mutually dependent

Question: Is Bob’s public key authentic for Alice?
Motivation: Credential Network

(1) Certificate Layer:

(2) Trust Layer:

- Both layers are mutually dependent
- Question: Is Bob’s public key authentic for Alice?
General Description

- Consider \( m \) layers \( L_i \), each specifying a certain property \( P_i \).
- Each layer \( L_i \) is represented by a graph \( G = (V, E_i) \).
- Evaluation of the system operation on layer \( L_i \) may require other layers, i.e. layers are not necessarily irreducible.
- Evaluation of system reliability involves all layers.
- In some cases, it is possible to reduce the layers into one single layer.
Consider $m$ layers $L_i$, each specifying a certain property $P_i$

Each layer $L_i$ is represented by a graph $G = (V, E_i)$

Evaluation of the system operation on layer $L_i$ may require other layers, i.e. layers are not necessarily irreducible

Evaluation of system reliability involves all layers

In some cases, it is possible to reduce the layers into one single layer
General Description

- Consider $m$ layers $L_i$, each specifying a certain property $P_i$
- Each layer $L_i$ is represented by a graph $G = (V, E_i)$
- Evaluation of the system operation on layer $L_i$ may require other layers, i.e. layers are not necessarily irreducible
- Evaluation of system reliability involves all layers
- In some cases, it is possible to reduce the layers into one single layer
Consider $m$ layers $L_i$, each specifying a certain property $P_i$.

Each layer $L_i$ is represented by a graph $G = (V, E_i)$.

Evaluation of the system operation on layer $L_i$ may require other layers, i.e., layers are not necessarily irreducible.

Evaluation of system reliability involves all layers.

In some cases, it is possible to reduce the layers into one single layer.
Consider $m$ layers $L_i$, each specifying a certain property $P_i$

Each layer $L_i$ is represented by a graph $G = (V, E_i)$

Evaluation of the system operation on layer $L_i$ may require other layers, i.e. layers are not necessarily irreducible

Evaluation of system reliability involves all layers

In some cases, it is possible to reduce the layers into one single layer
Representation
Formalization

Definition (Multilayer Reliability Network)

A Multilayer Reliability Network is a tuple

\[ M = (V, E_1, \ldots, E_m, s, t) \]

where

\[ V = \{ s^i, v_2, \ldots, v_{n-1}, t^i \} \] set of nodes, \( i = 1, \ldots, m \)

\[ E_i = \text{edge set w.r.t. layer } L_i, \ i = 1, \ldots, m \]

\[ s = (s^1, \ldots, s^m) \] vector of source nodes

\[ t = (t^1, \ldots, t^m) \] vector of terminal nodes
Joint Structure Function

- Evaluation of system reliability: compute reliability of joint-layer network
- Current assumptions:
  1. perfect nodes, independently failing edges
  2. networks are directed
  3. monotonicity
  4. network reliability: connectedness from \( s_i \) to \( t_i \)

**Definition (Joint Structure Function)**

A Joint Structure Function (JSF) is a Boolean function

\[
\Psi : \Omega^1 \times \ldots \times \Omega^m \rightarrow \{0, 1\}
\]

where \( \Omega^1, \ldots, \Omega^m \) are state spaces with respect to layers \( L^1, \ldots, L^m \), respectively.
Joint Structure Function

- Evaluation of system reliability: compute reliability of joint-layer network
- Current assumptions:
  1. perfect nodes, independently failing edges
  2. networks are directed
  3. monotonicity
  4. network reliability: connectedness from $s_i$ to $t_i$

Definition (Joint Structure Function)

A Joint Structure Function (JSF) is a Boolean function

$$\Psi : \Omega^1 \times \ldots \times \Omega^m \rightarrow \{0, 1\}$$

where $\Omega^1, \ldots, \Omega^m$ are state spaces with respect to layers $L^1, \ldots, L^m$, respectively.
Form and Properties of a general JSF

- Evaluate system reliability w.r.t. network property $P_i$ on layer $L_i$
- Then, the JSF $\Psi$ is defined as:

$$P_i(v_k) = \begin{cases} 
1 & \text{if } v_k = s^i, \quad \forall i \in L = \{1, \ldots, m\} \\
\bigvee_{l: e^i_{lk} \in I -(v_k)} \psi_i(\varphi_i(P_j(v_l)), \vartheta_i(e^j_{lk})) & \forall j \in J \subseteq L
\end{cases}$$

where

$$\psi_i = \varphi_i(P_j(v_l)) \land \vartheta_i(e^i_{lk})$$

and $P_i, \psi_i, \varphi_i, \vartheta_i$ are Boolean functions.

- Restriction to acyclic networks
Form and Properties of a general JSF

- Evaluate system reliability w.r.t. network property $P_i$ on layer $L_i$
- Then, the JSF $\Psi$ is defined as:

$$P_i(v_k) = \begin{cases} 1 & \text{if } v_k = s^i, \quad \forall \ i \in L = \{1, \ldots, m\} \\ \bigvee_{l: e_{lk}^j \in I-(v_k)} \psi_i[ \varphi_i(P_j(v_l)), \vartheta_i(e_{lk}^j) ], & \forall \ j \in J \subseteq L \end{cases}$$

where

$$\psi_i = \varphi_i(P_j(v_l)) \land \vartheta_i(e_{lk}^j) \quad \text{and}$$

$$P_i, \psi_i, \varphi_i, \text{ and } \vartheta_i \text{ are Boolean functions.}$$

- Restriction to acyclic networks
Simple Railway Example

1. Railtrack connectivity: $P_{rail}$
2. Power net connectivity: $P_{pwr}$
3. Train connection: $P_{conn}$

- Train connection from $v_1$ to $v_4$
- Evaluate JSF $P_{conn}$:

$$P_{conn}(v_1) = P_{pwr}(v_1) = P_{rail}(v_1) = 1$$

$$P_{conn}(v_4) = [P_{rail}(v_3) \land P_{pwr}(v_3) \land e_{34}^{rail} \land e_{34}^{pwr}]$$

$$\lor [P_{rail}(v_2) \land P_{pwr}(v_2) \land e_{24}^{rail} \land e_{24}^{pwr}]$$
Simple Railway Example

1. Railtrack connectivity: $P_{rail}$
2. Power net connectivity: $P_{pwr}$
3. Train connection: $P_{conn}$

- Train connection from $v_1$ to $v_4$
- Evaluate JSF $P_{conn}$:

\[
P_{conn}(v_1) = P_{pwr}(v_1) = P_{rail}(v_1) = 1
\]
\[
P_{conn}(v_4) = [P_{rail}(v_3) \land P_{pwr}(v_3) \land e_{34}^{rail} \land e_{34}^{pwr}] \lor [P_{rail}(v_2) \land P_{pwr}(v_2) \land e_{24}^{rail} \land e_{24}^{pwr}]
\]
\[
P_{rail}(v_3) = P_{rail}(v_1) \land e_{13}^{rail}
\]
\[
P_{pwr}(v_3) = P_{pwr}(v_1) \land e_{13}^{pwr}
\]
\[
P_{rail}(v_2) = P_{rail}(v_1) \land e_{12}^{rail}
\]
\[
P_{pwr}(v_2) = P_{pwr}(v_1) \land e_{12}^{pwr}
\]
Simple Railway Example

1. Railtrack connectivity: \( P_{\text{rail}} \)
2. Power net connectivity: \( P_{\text{pwr}} \)
3. Train connection: \( P_{\text{conn}} \)

- Train connection from \( v_1 \) to \( v_4 \)
- Evaluate JSF \( P_{\text{conn}} \):

\[
\begin{align*}
P_{\text{conn}}(v_1) &= P_{\text{pwr}}(v_1) = P_{\text{rail}}(v_1) = 1 \\
P_{\text{conn}}(v_4) &= \left[ P_{\text{rail}}(v_3) \land P_{\text{pwr}}(v_3) \land e_{34}^{\text{rail}} \land e_{34}^{\text{pwr}} \right] \\
&\lor \left[ P_{\text{rail}}(v_2) \land P_{\text{pwr}}(v_2) \land e_{24}^{\text{rail}} \land e_{24}^{\text{pwr}} \right]
\end{align*}
\]

\[
\begin{align*}
P_{\text{rail}}(v_3) &= P_{\text{rail}}(v_1) \land e_{13}^{\text{rail}} \\
P_{\text{pwr}}(v_3) &= P_{\text{pwr}}(v_1) \land e_{13}^{\text{pwr}} \\
P_{\text{rail}}(v_2) &= P_{\text{rail}}(v_1) \land e_{12}^{\text{rail}} \\
P_{\text{pwr}}(v_2) &= P_{\text{pwr}}(v_1) \land e_{12}^{\text{pwr}}
\end{align*}
\]
Outline

1. Introduction
2. Structure Function Representation
3. Network Reliability Evaluation
4. Multilayer Network Model
5. Conclusion
Knowledge representation based approach to network reliability evaluation: compilation of a reliability network into a PDAG

Polynomial time algorithm for generic SF representation: “choose” reliability problem using terminal selectors

Reliability computation based on cd-PDAG representation of the SF

Current and future work (selection):
- implementation
- elimination order heuristics
- comparison with other methods (e.g. OBDD)
- extend evaluation algorithm: apply to multilayer network model