ON COMBINING TRIPLE GRAPH GRAMMARS AND LINEAR OPTIMISATION TECHNIQUES

Erhan Leblebici, Anthony Anjorin, Andy Schürr
A Real-World Example: Overview

Robin Oppermann:
A Configurable, Model-Driven Approach to Optimal Scheduling using Triple Graph Grammars and Linear Programming.
Ongoing Master’s Thesis, Paderborn University in collaboration with dSpace
A Real-World Example: Overview

A series of recurring manual tests have to be executed.

Robin Oppermann:  
*A Configurable, Model-Driven Approach to Optimal Scheduling using Triple Graph Grammars and Linear Programming.*  
Ongoing Master’s Thesis, Paderborn University in collaboration with dSpace
A series of recurring manual tests have to be executed.

A test manager has to create a test schedule assigning test items to developers.

Robin Oppermann:  
A Configurable, Model-Driven Approach to Optimal Scheduling using Triple Graph Grammars and Linear Programming.  
Ongoing Master’s Thesis, Paderborn University in collaboration with dSpace
A Real-World Example: Overview

A series of recurring manual tests have to be executed.

A test manager has to create a test schedule assigning test items to developers.

Developers are only available for this purpose a few hours a week due to other responsibilities.

Robin Oppermann: 
*A Configurable, Model-Driven Approach to Optimal Scheduling using Triple Graph Grammars and Linear Programming.*
Ongoing Master’s Thesis, Paderborn University in collaboration with dSpace
A series of recurring manual tests have to be executed.

A test manager has to create a test schedule assigning test items to developers.

Developers are also on vacation now and then…

Developers are only available for this purpose a few hours a week due to other responsibilities.

Robin Oppermann:
A Configurable, Model-Driven Approach to Optimal Scheduling using Triple Graph Grammars and Linear Programming.
Ongoing Master’s Thesis, Paderborn University in collaboration with dSpace
A series of recurring manual tests have to be executed. A test manager has to create a test schedule assigning test items to developers. … and might not have the required expertise for all tasks. Developers are also on vacation now and then… Developers are only available for this purpose a few hours a week due to other responsibilities.

Robin Oppermann: A Configurable, Model-Driven Approach to Optimal Scheduling using Triple Graph Grammars and Linear Programming. Ongoing Master’s Thesis, Paderborn University in collaboration with dSpace
A Real-World Example: Metamodels

Area

Task

Execution

duration: EInt

Responsibility

Person

Availability

hours: EInt
week: WEEKS
A Real-World Example: Metamodels

There are predefined areas and people in charge of these areas.

- **Area**
- **Task**
- **Execution**
  - duration: EInt

- **Responsibility**
- **Person**
- **Availability**
  - hours: EInt
  - week: WEEKS
A Real-World Example: Metamodels

There are predefined areas and people in charge of these areas.

A test schedule maps executions to availabilities.
A Real-World Example: Allocation Rules

- t: Task
- p: Person
- e: Execution
- a: Availability

\[ e.\text{duration} \leq a.\text{hours} \]
A Real-World Example: Allocation Rules

In general, anyone can do anything as long as they have the time for it…
A Real-World Example: Allocation Rules

e.duration ≤ a.hours
A Real-World Example: Allocation Rules

Only test things that are not in your area of responsibility?
A Real-World Example: Allocation Rules

e.duration ≤ a.hours
A Real-World Example: Allocation Rules

Or perhaps being an expert makes you the best tester possible?
A Real-World Example: Allocation Rules

e.duration ≤ a.hours

```
<table>
<thead>
<tr>
<th>t:Task</th>
<th>p:Person</th>
</tr>
</thead>
<tbody>
<tr>
<td>e:Execution</td>
<td>a:Availability</td>
</tr>
<tr>
<td></td>
<td>++</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>ar:Area</th>
<th>r:Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>t:Task</td>
<td>p:Person</td>
</tr>
<tr>
<td>e:Execution</td>
<td>a:Availability</td>
</tr>
<tr>
<td></td>
<td>++</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>e':Execution</th>
<th>a':Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>t:Task</td>
<td>p:Person</td>
</tr>
<tr>
<td>e:Execution</td>
<td>a:Availability</td>
</tr>
<tr>
<td></td>
<td>++</td>
</tr>
</tbody>
</table>
```

```
e.duration ≤ a.hours
```

```
<table>
<thead>
<tr>
<th>ar:Area</th>
<th>r:Responsibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>t:Task</td>
<td>p:Person</td>
</tr>
<tr>
<td>e:Execution</td>
<td>a:Availability</td>
</tr>
<tr>
<td></td>
<td>++</td>
</tr>
</tbody>
</table>
```

```
e.duration ≤ a.hours
```
A Real-World Example: Allocation Rules

Should the same person test as many executions of the same task as possible? Or is this a terrible idea?
A Real-World Example: Allocation Rules

e.duration \leq a\.hours
Similar Application Domains
1. **Allocation Engineering:**
   - Tasks to resources
   - Programs to ECUs
   - Functions to nodes in a network
   - …
1. **Allocation Engineering:**
   - Tasks to resources
   - Programs to ECUs
   - Functions to nodes in a network
   - …

2. **Traceability Maintenance:**
   - Suggest traceability links
   - Check manually created traceability links
   - Flag “suspect links” after changes
Similar Application Domains

1. Allocation Engineering:
   - Tasks to resources
   - Programs to ECUs
   - Functions to nodes in a network
   - ...

2. Traceability Maintenance:
   - Suggest traceability links
   - Check manually created traceability links
   - Flag “suspect links” after changes

3. Model Synchronisation:
   - Start with existing, independently created models
   - Identify inconsistencies
Our Approach

2016:

Erhan Leblebici:
Towards a Graph Grammar-Based Approach to Inter-Model Consistency Checks with Traceability Support.
Bx@ETAPS 2016: 35-39

2017:

Erhan Leblebici, Anthony Anjorin, Andy Schürr:
Inter-model Consistency Checking Using Triple Graph Grammars and Linear Optimization Techniques.
FASE 2017: 191-207

2018:

Erhan Leblebici:
Inter-Model Consistency Checking and Restoration with Triple Graph Grammars.
PhD Thesis, Darmstadt University of Technology, Germany 2018

2018:

Nils Weidmann:
Consistency Management via a Combination of Triple Graph Grammars and Integer Linear Programming.
Master’s Thesis, Paderborn University, Germany 2018
Our Approach

2016:
Erhan Leblebici:
Towards a Graph Grammar-Based Approach to Inter-Model Consistency Checks with Traceability Support.
Bx@ETAPS 2016: 35-39

2017:
Erhan Leblebici, Anthony Anjorin, Andy Schürr:
Inter-model Consistency Checking Using Triple Graph Grammars and Linear Optimization Techniques.
FASE 2017: 191-207

2018:
Erhan Leblebici:
Inter-Model Consistency Checking and Restoration with Triple Graph Grammars.
PhD Thesis, Darmstadt University of Technology, Germany 2018

2018:
Nils Weidmann:
Consistency Management via a Combination of Triple Graph Grammars and Integer Linear Programming.
Master’s Thesis, Paderborn University, Germany 2018
Our Approach

Basic idea of how to perform consistency checking with TGGs

2016:
Erhan Leblebici: 
Towards a Graph Grammar-Based Approach to Inter-Model Consistency Checks with Traceability Support. 
Bx@ETAPS 2016: 35-39

2017:
Erhan Leblebici, Anthony Anjorin, Andy Schürr: 
Inter-model Consistency Checking Using Triple Graph Grammars and Linear Optimization Techniques. 
FASE 2017: 191-207

2018:
Erhan Leblebici: 
Inter-Model Consistency Checking and Restoration with Triple Graph Grammars. 
PhD Thesis, Darmstadt University of Technology, Germany 2018

2018:
Nils Weidmann: 
Consistency Management via a Combination of Triple Graph Grammars and Integer Linear Programming. 
Master’s Thesis, Paderborn University, Germany 2018
Our Approach

2016:
Erhan Leblebici: 
Towards a Graph Grammar-Based Approach for Inter-Model Consistency Checks with Traceability 
Bx@ETAPS 2016: 35-39

Full details, implementation, and evaluation in eMoflon

2017:
Erhan Leblebici, Anthony Anjorin, Andy Schürr: 
Inter-model Consistency Checking Using Triple Graph Grammars and Linear Optimization Techniques. 
FASE 2017: 191-207

2018:
Erhan Leblebici: 
Inter-Model Consistency Checking and Restoration with Triple Graph Grammars. 
PhD Thesis, Darmstadt University of Technology, Germany 2018

Nils Weidmann: 
Consistency Management via a Combination of Triple Graph Grammars and Integer Linear Programming. 
Master’s Thesis, Paderborn University, Germany 2018
Our Approach

2016:
Erhan Leblebici:
Towards a Graph Grammar-Based Approach to Inter-Model Consistency Checks with Traceability Support.
Bx@ETAPS 2016: 35-39

Erhan Leblebici, Anthony Anjorin, Andy Schürr:
Inter-model Consistency Checks using Triple Graph Grammars and Linear Optimization.
FASE 2017: 191-207

2017:
Erhan Leblebici:
Towards a Graph Grammar-Based Approach to Inter-Model Consistency Checks with Traceability Support.
Bx@ETAPS 2016: 35-39

Remaining formal proofs, industrial case with Siemens

2018:
Erhan Leblebici:
Inter-Model Consistency Checking and Restoration with Triple Graph Grammars.
PhD Thesis, Darmstadt University of Technology, Germany 2018

Nils Weidmann:
Consistency Management via a Combination of Triple Graph Grammars and Integer Linear Programming.
Master’s Thesis, Paderborn University, Germany 2018
Our Approach

2016:
Erhan Leblebici:
Towards a Graph Grammar-Based Approach to Inter-Model Consistency Checks with Traceability Support.
Bx@ETAPS 2016: 35-39

2017:
Erhan Leblebici, Anthony Anjorin, Andy Schürr:
Inter-model Consistency Checking Using Triple Graph Grammars and Linear Optimization Techniques.
FASE 2017: 191-207

2018:
Erhan Leblebici:
Inter-Model Consistency Checks with Traceability Support.
PhD Thesis, Darmstadt University

Nils Weidmann:
Consistency Management via a Combination of Triple Graph Grammars and Integer Linear Programming.
Master’s Thesis, Paderborn University, Germany 2018

Generalisation of the approach to other consistency management tasks (work in progress)
Step 1: Collect all Candidates
Step 1: Collect all Candidates

Use allocation rules (derived from a TGG) to create all possible correspondence links.
Step 1: Collect all Candidates

Use allocation rules (derived from a TGG) to create all possible correspondence links.
Step 1: Collect all Candidates

Use allocation rules (derived from a TGG) to create all possible correspondence links.

While these links are only candidates, they still make sense locally.
Step 1: Collect all Candidates

Use allocation rules (derived from a TGG) to create all possible correspondence links.

While these links are only candidates, they still make sense locally.

This step requires a necessary and sufficient condition for termination!
Step 1: Collect all Candidates

Use allocation rules (derived from a TGG) to create all possible correspondence links.

While these links are only candidates, they still make sense locally.

This step requires a necessary and sufficient condition for termination!
Step 2: Derive ILP
Step 2: Derive ILP
Step 2: Derive ILP

\[
\begin{align*}
\text{max} & \quad \vec{c} \cdot \vec{x} \\
A\vec{x} & \leq \vec{b}
\end{align*}
\]
Step 2: Derive ILP

\[
\begin{align*}
\text{max} & \quad \vec{c} \cdot \vec{x} \\
A \vec{x} & \leq \vec{b}
\end{align*}
\]

Which candidates will be part of the solution?

\[\vec{x} \in \mathbb{Z}_2^n\]
Step 2: Derive ILP

$$\max \; \vec{c} \cdot \vec{x}$$

$$A \vec{x} \leq \vec{b}$$

$$\vec{c} \in \mathbb{R}^n$$

Domain-specific weights for each candidate
Step 2: Derive ILP

\[ \max \vec{c} \cdot \vec{x} \]

E.g., prefer assigning multiple executions of the same task to the same person

\[ \vec{c} \in \mathbb{R}^n \]

Domain-specific weights for each candidate

\[ A\vec{x} \leq \vec{b} \]
Step 2: Derive ILP

\[
\text{max } \vec{c} \cdot \vec{x} \\
A \vec{x} \leq \vec{b}
\]

Constraints to ensure that the chosen solution is in the language of the TGG.
Step 2: Derive ILP

\[ \text{max } \overrightarrow{c} \cdot \overrightarrow{x} \]

\[ A \overrightarrow{x} \leq \overrightarrow{b} \]

- \( e': \) Execution
- \( a': \) Availability
- \( t: \) Task
- \( p: \) Person
- \( e: \) Execution
- \( e.\text{duration} \leq a.\text{hours} \)
Step 2: Derive ILP

\[
\begin{align*}
\text{max} & \quad \vec{c} \cdot \vec{x} \\
A \vec{x} & \leq \vec{b}
\end{align*}
\]

E.g., let’s assume all candidates for creating this link are: \(x_1, x_2, x_3\)
Step 2: Derive ILP

\[
\begin{align*}
\text{max } & \quad \overrightarrow{c} \cdot \overrightarrow{x} \\
A \overrightarrow{x} & \leq \overrightarrow{b}
\end{align*}
\]

E.g, let’s assume all candidates for creating this link are: \(x_1, x_2, x_3\)

This candidate requires at least one of them to exist:
\[
x_4 \Rightarrow x_1 \lor x_2 \lor x_3
\]
\[
x_4 \leq x_1 + x_2 + x_3
\]
Step 2: Derive ILP

\[
\max \quad \vec{c} \cdot \vec{x} \\
A \vec{x} \leq \vec{b}
\]

E.g, let’s assume all candidates for creating this link are:

\[x_1, x_2, x_3\]

All such inequalities are collected to form:

\[A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^n\]

This candidate requires at least one of them to exist:

\[x_4 \Rightarrow x_1 \lor x_2 \lor x_3\]
\[x_4 \leq x_1 + x_2 + x_3\]

e.duration \leq a.hours
Step 3: Solve (Optimise) ILP and Interpret Solution

\[
\begin{align*}
\text{max} & \quad \vec{c} \cdot \vec{x} \\
A\vec{x} & \leq \vec{b} \\
\vec{x} & \geq \vec{x}^* \\
\end{align*}
\]
Step 3: Solve (Optimise) ILP and Interpret Solution

\[
\text{max } \overrightarrow{c} \cdot \overrightarrow{x} \\
A\overrightarrow{x} \leq \overrightarrow{b}
\]

This step exploits mature ILP solvers
Step 3: Solve (Optimise) ILP and Interpret Solution

\[ \vec{x}^* \]
Step 3: Solve (Optimise) ILP and Interpret Solution
Step 3: Solve (Optimise) ILP and Interpret Solution

\[ \mathbf{x}^* \]
Step 3: Solve (Optimise) ILP and Interpret Solution

Our approach is tolerant in the sense that we can determine partial solutions (all variables are set to 0 in the worst case)
### Ongoing and Future Work

<table>
<thead>
<tr>
<th>Operation</th>
<th>Source</th>
<th>Corr</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>mark</td>
<td>create</td>
<td>mark</td>
</tr>
<tr>
<td>CO</td>
<td>mark</td>
<td>mark</td>
<td>mark</td>
</tr>
<tr>
<td>FWD_OPT</td>
<td>mark</td>
<td>create</td>
<td>create</td>
</tr>
<tr>
<td>BWD_OPT</td>
<td>create</td>
<td>create</td>
<td>mark</td>
</tr>
</tbody>
</table>
Our initial focus (Consistency Check via correspondence link creation)

<table>
<thead>
<tr>
<th>Operation</th>
<th>Source</th>
<th>Corr</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>mark</td>
<td>create</td>
<td>mark</td>
</tr>
<tr>
<td>CO</td>
<td>mark</td>
<td>mark</td>
<td>mark</td>
</tr>
<tr>
<td>FWD_OPT</td>
<td>mark</td>
<td>create</td>
<td>create</td>
</tr>
<tr>
<td>BWD_OPT</td>
<td>create</td>
<td>create</td>
<td>mark</td>
</tr>
</tbody>
</table>
# Ongoing and Future Work

<table>
<thead>
<tr>
<th>Operation</th>
<th>Source</th>
<th>Corr</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>mark</td>
<td>create</td>
<td>mark</td>
</tr>
<tr>
<td>CO</td>
<td>mark</td>
<td>mark</td>
<td>mark</td>
</tr>
<tr>
<td>FWD_OPT</td>
<td>mark</td>
<td>create</td>
<td>create</td>
</tr>
<tr>
<td>BWD_OPT</td>
<td>create</td>
<td>create</td>
<td>mark</td>
</tr>
</tbody>
</table>

- **Check Only**: Check existing triple for consistency
## Ongoing and Future Work

<table>
<thead>
<tr>
<th>Operation</th>
<th>Source</th>
<th>Corr</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>mark</td>
<td>create</td>
<td>mark</td>
</tr>
<tr>
<td>CO</td>
<td>mark</td>
<td>mark</td>
<td>mark</td>
</tr>
<tr>
<td>FWD_OPT</td>
<td>mark</td>
<td>create</td>
<td>create</td>
</tr>
<tr>
<td>BWD_OPT</td>
<td>create</td>
<td>create</td>
<td>mark</td>
</tr>
</tbody>
</table>

Normal initial (batch) fwd and bwd transformations; but now complete, tolerant, and optimal wrt. to an objective function.
All definitions, proofs, and most parts of the implementation can be formulated generically and configured for each case using the entries in this table!