Solverational Grammar and a Set of Evaluation Results

25.01.2010

Andreas Petter, Miroslav Zlatkov, Alexander Behring

TECHNISCHE UNIVERSITÄT DARMSTADT

25.01.2010

Technical Report No. TUD-CS-2010-017
Technische Universität Darmstadt

Telecooperation Report No. TR-12,
The Technical Reports Series of the TK Research Division, TU Darmstadt
ISSN 1864-0516
http://www.tk.informatik.tu-darmstadt.de/de/publications/

Contents

1 Solverational Grammar

Parts of Solverational were presented in [10, 8, 7, 9, 11, 12]. This technical report introduces the grammar - including the extension of optimization - on a technical level for reference, only (benefits etc. are omitted).

The major benefits of using Solverational are:

- Constraint Solving
- Optimization
- Derivate Problems

While constraint solving helps in implementing transformations with constraints in the target model, optimization selects the "best" target model from a set of possible models. This is useful when, e.g. non-confluent transformation shall be confluent. Then an appropriately defined target function selects one model out of a set of models and makes the transformation confluent.

Derivate problems help if the transformation developer wants to provide several options for classes of target model elements. Then a target model element can be of different classes (i.e. types) when the transformation is finished. This is especially useful in combination with a target function which selects the best classes for model elements. Also if constraint solving is used in the transformation, a derivate problem can select an option for a model element attached to constraints, where the constraint system is feasable (others may not be feasable).

The grammar of Solverational is comprised of three parts:

1. The abstract syntax
2. The concrete textual syntax
3. The concrete graphical syntax

While both of the concrete syntaxes are used to formalize model-to-model transformations the abstract syntax is mostly used to represent the transformation as a model.

1.1 Abstract Syntax

Since Solverational is a modification of QVT Relations [6] and the OCL syntax [5] the abstract syntax is heavily based on the original grammar. In fact we only made minor changes to the syntax. For presentation purposes we copied the original figures from the QVT Relations document (based on the refined QVT Relations syntax given in the document published on 07.07.2007 [6]) and present our changes within the abstract syntax presented by the OMG. The abstract syntax of QVT Relations is an extension to Essential OCL [5] and EMOF [3]. It is comprised of three packages:

1. The QVT Base package
2. The QVT Template package
3. The QVT Relations package

We present the extensions we made to each package to extend the syntax to the Solverational language.

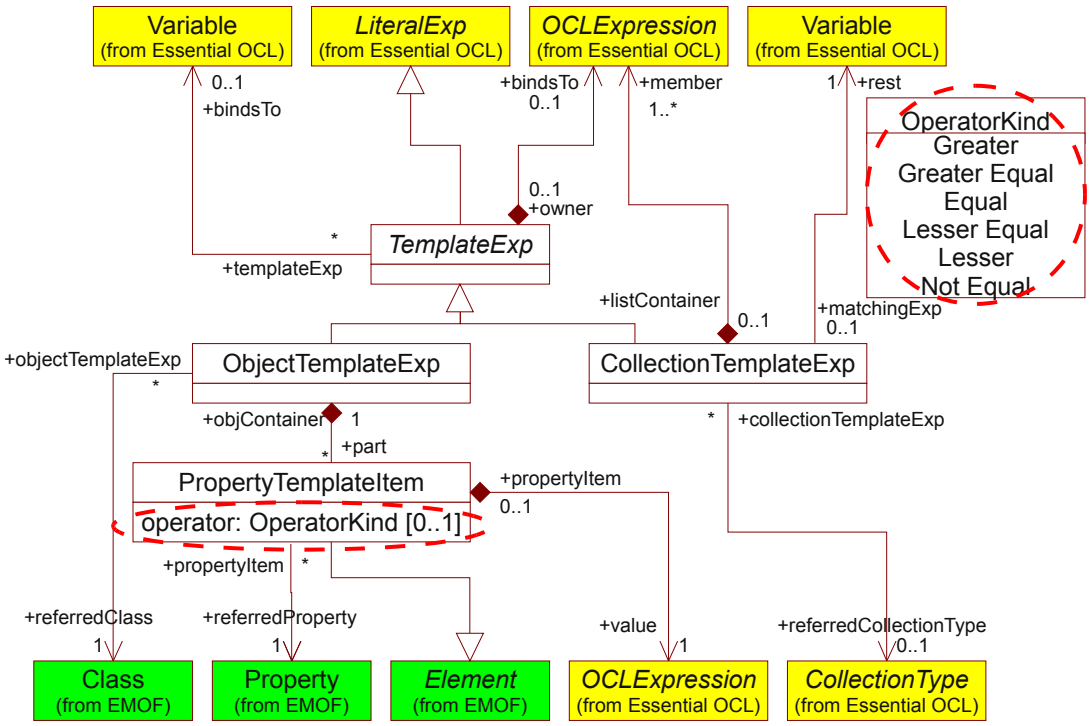

Figure 1: Extensions to QVT Template Package.

1.1.1 QVT Base Package

The QVT Base package has not been changed.

1.1.2 QVT Template Package

In the QVT Template package we added support for different comperators. In the following figures we marked the changes with a red broken line. One can see that for "PropertyTemplateItems" we introduced an attribute called "operator" which contains the comparator out of "OperatorKind" (see Figure 1). If the attribute is left out the "equal" comparator will be used by default.

These comparators introduce inequalities. Therefore, with Solverational it is possible to declare constraint solving problems over attribute values.

1.1.3 QVT Relation Package

The QVT Relation Package has been extended such that target functions may be declared in "RelationalTransformation" elements as an attribute called "optimization" (see Figure 2). This attribute associates to an "OptFunction" model element, which associates to an OCL Expression which, can be minimized (attribute "maximize" is false) or maximized ("maximize" is true).

The introduction of derivate problems is done using alternative domains, which is shown in a blue broken oval in Figure 2. An alternative domain is alternative to all other domains

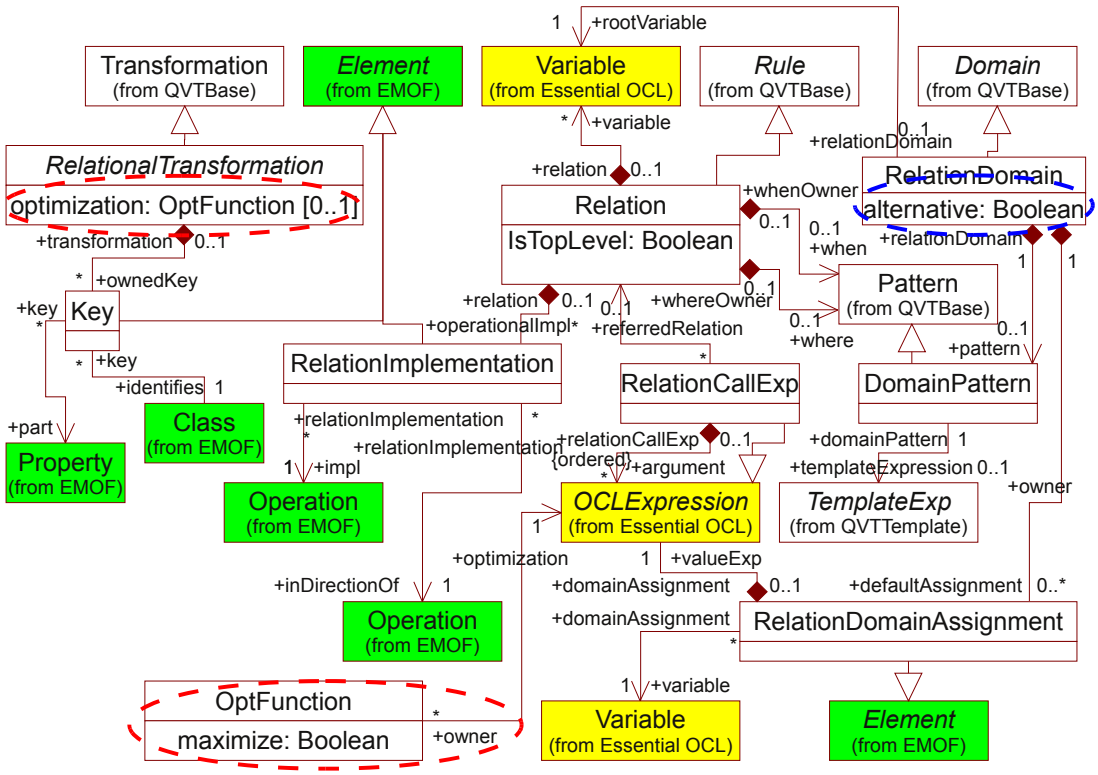

Figure 2: Extensions to QVT Relation Package.

which are attached to the same model descriptor in the same "RelationalTransformation". The different classes come into play if several target domains are alternative. Each alternative may be associated to a different class, which in fact then are alternatives in the model generation process. The PropertyTemplateItems are only valid for each alternative (of course they may be duplicated, but still each duplicate is valid only in its domain context).

1.2 Concrete Syntax: Textual

The changes we made to the textual syntax directly reflect the changes we made to the abstract syntax. Therefore we just present the changes and let readers recall the changes listed in the abstract syntax above.

Introduction of constraints:
We change the definition of the non-terminal symbol "propertyTemplate":

```
<propertyTemplate> ::= <identifier>
                       ('<' | '<=' | '=' | '>=' | '>' | '<>')
                       <OclExpressionCS>
```

Additionally, for Solverational it is absolutely allowed and encouraged, that on the right side of PropertyTemplates variables and attributes may not be assigned. This is a feature introduced by

constraint solving, and can currently not be handled by any other QVT Relations transformation engine (variables just get assigned but attributes do not get enforced).

An identifier may be used multiple times on the right side of a propertyTemplate for a given domain. This makes sense in the context of constraint solving as attributes may be constraint several times, e.g. by defining an upper or a lower bound for the attribute.

Introduction of optimization functions:

To introduce we added a non-terminal symbol "optFunction" to the concrete synatx (it works similar to the one in the abstract syntax, see above):

```
<optFunction> ::= ('maximize' | 'minimize')
    <OclExpressionCS> ';'
```

Additionally we changed the definition of the non-terminal symbol "transformation":

```
<transformation> ::= 'transformation' <identifier>
    '(' <modelDecl> (',' <modelDecl>)* ')'
    ['extends' <identifier>]
    '{' [<optFunction>] <keyDecl>*
    ( <relation> | <query> )* '}'
```

Thereby we can define a single target function for each transformation.

Extension to achieve derivate problems:

To achieve derivate problems we just added a key-word "alternative" to the concrete syntax which can be used when defining domains:

```
<domain> ::= [<checkEnforceQualifier>] ['alternative'] 'domain'
    <modelId> <template>
    ['implementedby' <OperationCallExpCS>]
    ['default_values' '{' (<assignmentExp>)+ '}'] ';'
```

1.3 Concrete Syntax: Graphical

For the graphical syntax we needed to extend the UML object diagrams [4] which are mentioned in the QVT specification being the basis for the graphical syntax.

Therefore, we extended the "Slot" with the same "operator" attribute as we did in the abstract syntax for Solverational (see Figure 3). Similarly we added the "OperatorKind" enumeration to eidentify the comparators.

To introduce optimization we added the possibility to add target functions into the center of the transformation symbol (see Figure 4).

For the introduction of derivate problems, nothing needs to be done. For the QVT Relation graphical syntax we just extend the option to use several domains on each side of the transformation symbol. This results in a graphical syntax for alternative domains.

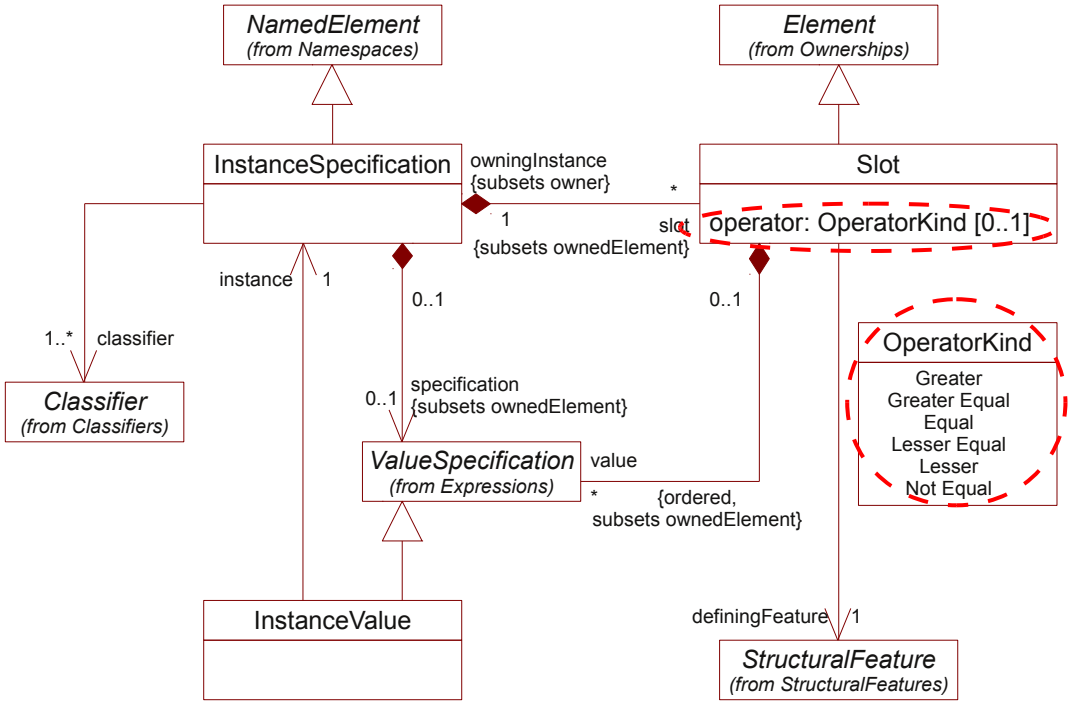

Figure 3: Extensions to UML Object Diagrams.

2 Comparison of Solverational and SGG

In the following we look at the spatial relationships supported by Spatial Graph Grammars (SGG) and describe what set of constraints could be used in Solverational in order to represent them. In this way we aim to show that the functionalities of Solverational for representing graphical user interfaces are at least as good as these of SSG.

2.1 Spatial Relationships

The spatial relationships considered here are taken from chapter 3 in the article *Spatial Graph Grammars for Graphical User Interfaces* by Kong et al. [2]. We assume that all objects are rectangles unless it is stated otherwise. This is not a serious drawback as most of the objects in graphical user interfaces have rectangular shapes. Even though it is not stated explicitly, Kong et al. [2] use only rectangular objects in their examples as well.
The set of all objects is denoted by $\mathcal{O}$. Coordinates increase to the bottom and the right. Thus, for a (rectangular) object $a \in \mathcal{O}$ with positive width and height, $a.w$ and $a.h$ respectively, if the coordinates of its upper left corner are $(a.x, a.y)$, then these of its lower right corner are $(a.x + a.w, a.y + a.h)$. Figure 5 depicts the coordinates of important points of a rectangular object.

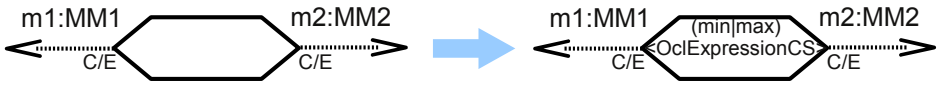

Figure 4: Introduction of target functions to the graphical syntax.

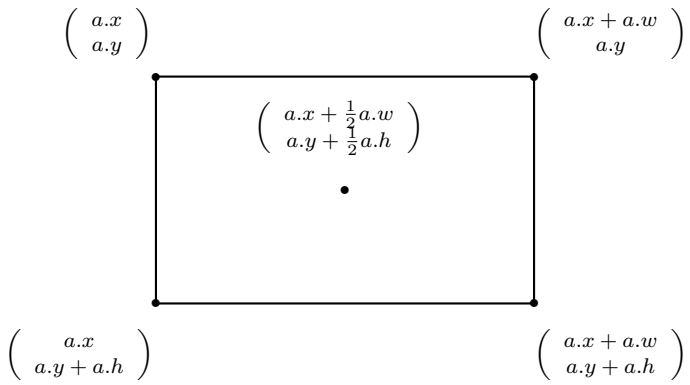

Figure 5: Important points of a rectangular object.

2.1.1 Topology

For a graphical example of the topological relationships see figure 4 in the article of Kong et al. [2].

Touch: a touches $b \iff$
$[(a.x + a.w = b.x) \wedge (a.y + a.h \geq b.y) \wedge (a.y \leq b.y + b.h)] \vee$
$[(a.x = b.x + b.w) \wedge (a.y \leq b.y + b.h) \wedge (a.y + a.h \geq b.y)] \vee$
$[(a.x + a.w \geq b.x) \wedge (a.x \leq b.x + b.w) \wedge (a.y + a.h = b.y)] \vee$
$[(a.x \leq b.x + b.w) \wedge (a.x + a.w \geq b.x) \wedge (a.y = b.y + b.h)]$

In: a is in $b \iff$
$(a.x \geq b.x) \wedge (a.y \geq b.y) \wedge (a.x + a.w \leq b.x + b.w) \wedge (a.y + a.h \leq b.y + b.h)$

Overlap: a overlaps $b \iff$
$(a.x < b.x + b.w) \wedge (a.y < b.y + b.h) \wedge (a.x + a.w > b.x) \wedge (a.y + a.h > b.y)$

Disjoint: a is disjoint with $b \iff$
$(a.x > b.x + b.w) \vee (a.y > b.y + b.h) \vee (a.x + a.w < b.x) \vee (a.y + a.h < b.y)$

Cross: We consider this relationship last because it is the only one where one of the objects is of one less dimension than the other one and where Kong et al. [2] explicitly use a line segment. In Solverational one can determine a line segment in a way similar as with a rectangle: one needs the coodinates $(a.x, a.y)$ and $(a.x + a.w, a.y + a.h)$ of the end points of the segment, but here $a.h$ could be nonpositive in order to allow for all possible slopes of the line segment (see figure 6 for an example.)
To explain the idea behind the derivation of the constraints for the topological relation

"cross", we need to introduce two auxilliary structures: given a line segment a, we will use the line l_a and the rectangular hull $\bar{a}$ containing the line segment. These are depicted on figure 6 and are defined as

$$l_a := \{(x_l, y_l) \in \mathbb{R}^2 | (a.h)(x_l - a.x) + (-a.w)(y_l - a.y) = 0\}$$

and

$$\bar{a} := \{(x_r, y_r) \in \mathbb{R}^2 | [a.x \leq x_r \leq a.x + a.w] \wedge [(a.y \leq y_r \leq a.y + a.h) \vee (a.y + a.h \leq y_r \leq a.y)]\}.$$

Endowed with these two structures one could show that for a line segment a and a rectangle b

$$a \text{ is cross with } b \Longleftrightarrow \{l_a \text{ goes through } b\} \wedge \{\bar{a} \text{ overlaps } b\}.$$

The translation of the relation "overlap" into Solverational constraints has already been shown. The condition $\{l_a$ goes through $b\}$ is equivalent to

$$[\langle (a.h, -a.w), (b.x, b.y) - (a.x, a.y) \rangle > 0] \wedge [\langle (a.h, -a.w), (b.x + b.w, b.y + b.h) - (a.x, a.y) \rangle < 0] \text{ if } a.h \leq 0$$

or

$$[\langle (a.h, -a.w), (b.x + b.w, b.y) - (a.x, a.y) \rangle > 0] \wedge [\langle (a.h, -a.w), (b.x, b.y + b.h) - (a.x, a.y) \rangle < 0] \text{ if } a.h \geq 0$$

where $\langle ., . \rangle$ denotes the scalar product. With these conditions, which guarantee that there are corners of the rectangle b lying in both half-planes of l_a, one obtains the result

the line segment a is cross with the rectangle $b \Longleftrightarrow$

$$\begin{aligned}
&\{[(a.h)(b.x) + (a.h)(b.w) - (a.h)(a.x) - (a.w)(b.y) - (a.w)(b.h) + (a.w)(a.y) > 0] \wedge \\
&[(a.h)(b.x) - (a.h)(a.x) - (a.w)(b.y) + (a.w)(a.y) > 0] \wedge \\
&[a.x < b.x + b.w] \wedge [a.x + a.w > b.x] \wedge [a.y > b.y] \wedge \\
&[a.y + a.h < b.y + b.h] \wedge [a.h \leq 0]\} \\
&\vee \\
&\{[(a.h)(b.x) + (a.h)(b.w) - (a.h)(a.x) - (a.w)(b.y) + (a.w)(a.y) > 0] \wedge \\
&[(a.h)(b.x) - (a.h)(a.x) - (a.w)(b.y) - (a.w)(b.h) + (a.w)(a.y) > 0] \wedge \\
&[a.x < b.x + b.w] \wedge [a.x + a.w > b.x] \wedge [a.y + a.h > b.y] \wedge \\
&[a.y < b.y + b.h] \wedge [a.h \geq 0]\}
\end{aligned}$$

2.1.2 Direction

To describe direction relations the SGG approximates objects as points and adopts the cone-based approach of Peuquet et al. [13] Approximating rectangular objects by their centers, the two direction relations given as an example by Kong et al. in [2] could be specified in Solverational as follows:

- For $dir_4 : \mathcal{O} \times \mathcal{O} \longrightarrow D_4 := \{\mathrm{N}, \mathrm{E}, \mathrm{S}, \mathrm{W}\}$,
  set
  $\bar{x} = b.x + 0.5 * (b.w) - a.x - 0.5 * (a.w)$ and
  $\bar{y} = b.y + 0.5 * (b.h) - a.y - 0.5 * (a.h)$.
  Then
  $dir_4(a, b) = N \Longleftrightarrow (abs(\bar{y}) \geq \mathrm{abs}(\bar{x})) \wedge (\bar{y} \leq 0)$,
  $dir_4(a, b) = E \Longleftrightarrow (\mathrm{abs}(\bar{y}) < \mathrm{abs}(\bar{x})) \wedge (\bar{x} > 0)$,

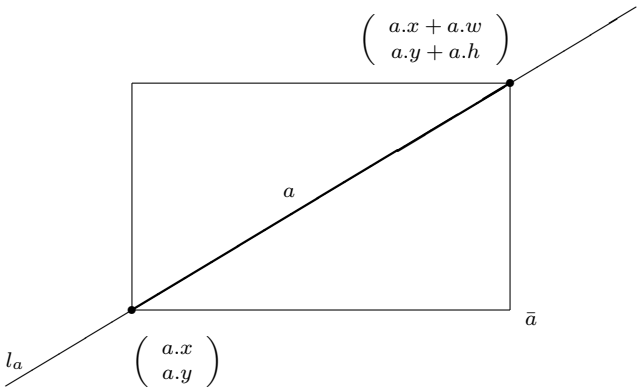

Figure 6: Important points of a line segment.

$dir_4(a,b) = W \iff (\text{abs}(\bar{y}) < \text{abs}(\bar{x})) \wedge (\bar{x} < 0)$ and
$dir_4(a,b) = S \iff (\text{abs}(\bar{y}) \geq \text{abs}(\bar{x})) \wedge (\bar{y} > 0)$.

- For $dir_8 : \mathcal{O} \times \mathcal{O} \longrightarrow D_8 := \{\text{N}, \text{NE}, \text{E}, \text{SE}, \text{S}, \text{SW}, \text{W}, \text{NW}\}$,
  set
  $\bar{x} = b.x + 0.5 * (b.w) - a.x - 0.5 * (a.w)$ and
  $\bar{y} = b.y + 0.5 * (b.h) - a.y - 0.5 * (a.h)$.
  Then
  $dir_8(a,b) = N \iff (\bar{y} \leq -d_2 * \text{abs}(\bar{x}))$,
  $dir_8(a,b) = E \iff (\bar{y} > -d_1 * \bar{x}) \wedge (\bar{y} \leq d_1 * \bar{x}) \wedge (\bar{x} > 0)$,
  $dir_8(a,b) = W \iff (\bar{y} > d_1 * \bar{x})) \wedge (\bar{y} \leq -d_1 * \bar{x}) \wedge (\bar{x} < 0)$,
  $dir_8(a,b) = S \iff (\bar{y} > d_2 * \text{abs}(\bar{x}))$,
  $dir_8(a,b) = NE \iff (\bar{y} > -d_2 * \bar{x}) \wedge (\bar{y} \leq -d_1 * \bar{x})) \wedge (\bar{x} > 0)$,
  $dir_8(a,b) = NW \iff (\bar{y} > d_2 * \bar{x}) \wedge (\bar{y} \leq d_1 * \bar{x}) \wedge (\bar{x} < 0)$,
  $dir_8(a,b) = SE \iff (\bar{y} > -d_1 * \bar{x}) \wedge (\bar{y} \leq -d_2 * \bar{x}) \wedge (\bar{x} < 0)$ and
  $dir_8(a,b) = SW \iff (\bar{y} > d_1 * \bar{x}) \wedge (\bar{y} \leq d_2 * \bar{x}) \wedge (\bar{x} > 0)$,
  where d_1 and d_2 are numerical approximations of $\tan(\frac{\pi}{8})$ and $\tan(\frac{3\pi}{8})$ respectively.

As in the second example, with the help of different numerical approximations one could also specify other (more complex) cone-based direction relations.

2.1.3 Distance

For a given distance granularity $Q = \{q_0, q_1, \ldots, q_n\}$ of increasing totaly ordered distance distinctions q_i with their associated distance ranges δ_i $(i = 0, \ldots, n)$ we make the assumption that δ_i could be described by an interval. That is,

$(\forall \delta_i)(\exists l_i, u_i \in \mathbb{R} \cup \{\infty\} \mid 0 \leq l_i \leq u_i)$
$\quad \delta_i = (l_i, u_i) \vee \delta_i = [l_i, u_i) \vee \delta_i = (l_i, u_i] \vee \delta_i = [l_i, u_i]$.

This assumption is not very restrictive because many more complex distance ranges could be defined as union of such intervals. Under this assumption

$$\begin{aligned}
(\forall q_i \mid \delta_i = (l_i, u_i))(\forall a, b \in \mathcal{O}) \qquad & aq_ib \iff l_i < d(a,b) \wedge d(a,b) < u_i \\
(\forall q_i \mid \delta_i = [l_i, u_i))(\forall a, b \in \mathcal{O}) \qquad & aq_ib \iff l_i \leq d(a,b) \wedge d(a,b) < u_i \\
(\forall q_i \mid \delta_i = (l_i, u_i])(\forall a, b \in \mathcal{O}) \qquad & aq_ib \iff l_i < d(a,b) \wedge d(a,b) \leq u_i \\
(\forall q_i \mid \delta_i = [l_i, u_i])(\forall a, b \in \mathcal{O}) \qquad & aq_ib \iff l_i \leq d(a,b) \wedge d(a,b) \leq u_i
\end{aligned}$$

where $d(a, b)$ is set to be the distance between a and b

$$d(a,b) = [(b.x + 0.5 * b.w - a.x - 0.5 * a.w)^2 + (b.y + 0.5 * b.h - a.y - 0.5 * a.h)^2]^{0.5}$$

2.1.4 Alignment

In the following we consider only alignment in the horizontal direction. The constraints for vertical alignment can be derived from the ones for horizontal alignment by replacing the attributes $.x$ and $.w$ with $.y$ and $.h$ respectively for every object. For a graphical example of the alignment relationships see figure 6 in the article of Kong et al. [2].

Gap: There is a gap between a and $b \iff (a.x + a.w < b.x) \vee (a.x > b.x + b.w)$

Meet: a meets $b \iff (a.x + a.w = b.x) \vee (a.x = b.x + b.w)$

Equal: a equals $b \iff (a.x = b.x) \wedge (a.x + a.w = b.x + b.w)$

Interleaving: a interleaves with $b \iff$
$[(a.x < b.x) \wedge (a.x + a.w > b.x) \wedge (a.x + a.w < b.x + b.w)] \vee$
$[(a.x > b.x) \wedge (a.x > b.x + b.w) \wedge (a.x + a.w > b.x + b.w)]$

Middle: a is in the middle of $b \iff (a.x > b.x) \wedge (a.x + a.w < b.x + b.w)$

Left (or Top): a and b are left aligned $\iff (a.x = b.x)$

Right (or Bottom): a and b are right aligned $\iff (a.x + a.w = b.x + b.w)$

2.1.5 Spatial Granularity and Spatial Hierarchy

Spatial Granularity: Different levels of granularities for the distance and direction relationships are supported in Solverational as well. This is done by defining more detailed ranges or directions model respectively. Different granularities for the topological and alignment relationships could be obtained by defining spatial hierchies, see next paragraph.

Spatial Hierarchy: Relationship $\bar{R}$ preserves relationship R if whenever R holds for two objects a and b, $\bar{R}$ holds for them as well. This implies that the constraints of R are a subset of the constraints of $\bar{R}$:

$$(\text{constraints of } \bar{R}) \equiv [(\text{constraints of } R) \wedge \ldots]$$

Developers can thus implement spatial hierarchies in Solverational by reusing the constraints of a relationship that is being preserved.

2.2 Using the constraints in a transformation. An example.

As an example how to implement the results from our comparison we present a transformation for the relation "gap".

```
transformation t(a:inputMM, c:outputMM) {
    relation relationGap {
        domain a Node {
            gap = b:Node {}
        };
        alternative domain c NewNode {
            c.x < c.w + d.x,
            gap = d:NewNode {}
        };
        alternative domain c NewNode {
            c.x > d.x + d.w,
            gap = d:NewNode {}
        };
    }
}
```

2.3 Interpretation

In this section we presented a set of constraints which represent all spatial relations given in [2]. Thereby, we showed that all relations can be implemented using Solverational.

This has two implications:

1. the expressiveness of Solverational concerning the relations is as good as with SGGs
2. Solverational is additionally able to handle constraints and optimization

However, with SGG it is possible to achieve a polynominal time sub-graph search. This is beneficial but, the set of possible graphs which can be is restricted to a set of graphs called "selection-free-graphs". According to [2] it is not clear how many graphs fall into this category. Even simple graphs, like fully meshed graphs cannot be implemented using SGGs and polynominal time. For all other graphs the runtime of sub-graph searches for SGGs is unknown, too as it is for Solverational.

Additionally with Solverational it is possible to implement constraint solving and optimization into the graph transformation process for e.g. implementing Fitt's Law or its descendents. This is beneficial for the transformation of user interface models.

Comparing both languages we can conclude that SGGs are faster for certain graphs but Solverational implements all relations of SGGs plus Solverational has capabilities for constraint solving and optimization.

3 Evaluation of Appropriateness

Although it is an all purpose transformation language, Solverational was designed having model transformation of user interface models in mind.

Option	Code
n/a	0
can be implemented using Solverational	1a
can be implemented using Solverational but making several assumptions	1b
cannot be implemented using Solverational	1c
can be implemented using a standard transformation language	2a
can be implemented using a standard transformation language but making several assumptions	2b
cannot be implemented using any transformation language	2c

Table 1: Evaluation Options

Starting from there, we studied how well Solverational fits the needs of transforming user interface models. To perform the evaluation two experts examined the Apple Human Interface Guidelines [1]. Each guideline represents a rule that could be implemented in Solverational. For each guideline we examined if it is possible to implement it using Solverational. Table 1 shows the options the experts could choose for each rule. It can either be implemented using Solverational (1a), a standard model transformation (2a), implemented having some assumptions in mind (1b for Solverational,2b standard transformation languages) or cannot be implemented (1c,2c). Rules like "Your product development team should include a skilled writer who is responsible for reviewing all user-visible onscreen text as well the instructional documentation. The writer should refer to the Apple Publications Style Guide for guidance on Apple-specific terminology." are out of scope (0).

Code	Sum	%, incl. 0	%, excl. 0
0	34	24.46%	
1a	10	7.19%	9.52%
1b	4	2.88%	3.81%
1c	3	2.16%	2.86%
2a	39	28.06%	37.14%
2b	24	17.27%	22.86%
2c	25	17.99%	23.81%

Table 2: Evaluation Results

Table 2 presents the overall countings, while table 3 shows the results for each individual rule.

Each rule is given with the page number of the style guide, an idetifier for cross references, the Headline of the section where the rule can be found, the wording of the rule as it can be found in the style guide, the result, and a short description of how this rule could be implemented or not. In most cases we assumed several models to exist. Most notably there are three models: a task model, an abstract user interface model and a concrete user interface model. Additionally there are several other models providing information to the transformation process, such as a

configuration model, a file formats and contents model, a standard directories model (part of the configuration model), a workflow model, a cursor model (considered to be part of the concrete user interface model), and a key model (considered to be part of the concrete user interface model).

The results are summerized in Table 2. The left-most column shows the code of the result, as it has been used in Table 3.

3.1 Interpretation

The results of this study are quite promising. More than 50% (55.4%) of the rules can be translated to a formal representation using Solverational. Thereby, 10 rules can only be implemented using Solverational because constraint solving or optimization are required. An additional 4 rules can only be implemented using Solverational by making assumptions about the structure of the information given in the rule. On the other hand three rules can not be implemented in Solverational, but might be implementable using a different model-to-model transformation language. This is because Solverational does not support features like string processing. Even counting these rules: with a standard model-to-model transformation language less than 50% (47.5%) can be implemented. Therefore, it seems that it might be better to choose Solverational as the language of choice when it comes to selecting a model-to-model transformation language to implement rules of a style guide. We were somehow surprised, that the benefit of using Solverational comes only from 8% of the rules.

Regarding the reuslts many of the rules which can not be translated into a formal representation are more issues like the organization of the developement team or parameters of the project setup in an IDE. Therefore these are out of scope. If this is taken into account almost 75% of the rules can be implemented using Solverational.

However, there are some threats to validity. To some degree, the results of the study are not objective, because they represent the individual interpretation of the guidelines by two experts, only. Furthermore, the reliability of the study can not be guaranteed as the guidelines are partially written very vague and allow for different interpretations. Therefore, performing the study even with the same two experts might result in slightly different results when re-executing the study.

On the other hand it is the first study performed on this topic in this way and the results are quite promising, because more than 50% of the rules can be implemented using a model-to-model transformation language, called Solverational.

Page	Id	Headline	Rule	Res	How
61	1	The Always-On Environment	Avoid relying on a restart to get rid of cached or temporary files that may use up disk space. Be prepared to remove these files yourself when they are no longer needed.	2c	must be avoided by the application developer, as deleting files is a feature implemented in code
	2		Avoid relying on startup or login items to initiate user-level processes. If the user quits a process initiated only at boot time, that process will be unavailable until the machine restarts.	2c	must be avoided by the application developer, as the application start is a feature implemented by project setups etc.
	3		Avoid requiring users to reboot as a part of an installation or software update unless absolutely necessary. Your application is probably not the only one they have open, so a restart can come as a rude interruption.	2c	must be avoided by the application developer, as rebooting is a feature implemented in code
	4	Displays	Avoid making assumptions about display size.	2a	depending on the toolkit used by the application, which would be Cocoa for Apple Mac OS X
	5		You should avoid making assumptions about the aspect ratio.	2a	depending on the toolkit used by the application, which would be cocoa for Apple Mac OS X
62	6	The Dock	New windows should not overlap the boundaries of the Dock.	1a	given the bounds of the dock in advance constraint solving could be used
	7		You should prevent users from moving or resizing windows so that they are behind the Dock.	2c	must be implemented in application code
	8		Use badging to convey status information in an unobtrusive manner.	1a	if it is clear which information needs to be badged, a constraint can be added which shows the information in the right upper corner of the icon
	9		Use the Notification Manager to convey more serious information, such as error conditions.	2a	if the information is given in two parts in a model the transformation can split and assign it to the appropriate places such as the information manager or the alert box/window.

Page	Id	Headline	Rule	Res	How
	10		Clicking an application icon in the Dock should always result in a window becoming active.	2a	transformation of icons into actions of a task diagram
	11		If the application is not open, a new window should open. In a document-based application, the application should open a new, untitled window. In an application that is not document-based, the main application window should open.	2a	appropriate transformations of icons into appropriate actions of a task diagram
	12		When a user clicks an open application's icon in the Dock, the application becomes active and all open unminimized windows are brought to the front; minimized document windows remain in the Dock. If there are no unminimized windows when the user clicks the Dock icon, the last minimized window should be expanded and made active.	2c	must be implemented in application code
	13		Control-clicking an application icon in the Dock displays a menu that allows users to perform tasks such as quitting the application, hiding it, or showing its location in the file system.	2a	transformation of possible tasks in a task diagram into an appropriate context menu for the application icon
63	14	The Finder	Make sure your application bundle has a .app extension.	0	it is pointless to implement that in a transformation, as this is automatically done by IDEs
	15		Package CFM applications in a bundle.	0	this is done by the project setup in the IDE
	16		Use an information property list to communicate information to the Finder.	2a	the information could be automatically extracted from given models (by using special attributes for markings) and the list could be generated using a transformation

Page	Id	Headline	Rule	Res	How
	17		When saving files of your own document types, be sure to give them appropriate filename extensions to ensure platform interoperability and to support the Mac OS X user experience.	2a	if the save dialog is modelled with an attribute for the file extension the transformation could use the name of the application to implement that
	18		Avoid changing the creator type of existing documents.	2c	must be avoided by the application developer, as changing the creator type is a feature implemented in code
	19	File Formats and Filename Extensions	Whenever possible, include support for industry-standard file formats in your documents.	2a	there could be a model element which contains all file formats. Then it could be handed over to load and save dialogs as above in rule id 17
	20		When saving user-configurable data, make sure you store it in a plain-text file that the user can modify.	1c	saving configuration files is usually implemented in code
	21		Applications that save documents should be sure to include a filename extension appropriate to the contents of the document.	2a	the content of the document is mostly given by the type of the application which could also be modelled. The transformation adds the attribute to the save dialog as in rule id 17
	22		Applications should take care to respect the user's filename extension preferences when displaying the names of files and documents.	2c	as this is a user preference it cannot be determined at design time
64	23	Internationalization	Implement your program as a bundle	0	done by the IDE
	24		Support Unicode text.	0	to be implemented in code
	25		Modify your code to get user-visible strings from .strings files.	0	subject to the developer
	26		Use nib files to store your user interface data.	0	handled by the modelling environment
	27	Multiple User Issues	Shared memory, cache files, semaphores, and named pipes must be carefully labeled to prevent corruption by users running the same application in different sessions.	2c	this is subject to the code of the application, as semaphores are usually not modelled (except for verification, which is performed seldomly)

Page	Id	Headline	Rule	Res	How
	28		Applications cannot assume that they have exclusive access to any system resources, such as a CD or DVD drive.	1c	this is subject to the code of the application, as semaphores are usually not modelled (except for verification, which is performed seldomly)
	29		Named resources that might potentially be accessible to an application from multiple user sessions should incorporate the session ID into the name of the resource.	2c	the session id is only available at runtime, therefore it cannot be handled in a design time transformation
65	30		Users on a computer can include both local and network users, so do not assume a user's home directory is on a local volume.	2a	the location of the home directory can be modelled as an abstract model element and the transformation only uses the abstract element if it is instructed to add paths to a list element
	31	Resource Management	Include all required resources inside your application bundle.	0	done by the IDE
	32		Include only the specific subset of files that require localization in your application bundle's language-specific resource directories. If a resource does not require localization, there is no need to create extra copies of it.	0	done by hand or an internationalization toolkit provided by the IDE
	33		Use an installer to place optional resources in the appropriate Library subdirectory of the user's system. Optional resources are things like document templates or other resources that are useful to an application but not required for it to launch. Most application-related files should go in an application-specific subdirectory of /Library/Application Support or /Library/Application Support.	0	done by hand or an installer toolkit provided by the IDE
	34		Avoid storing data in the resource fork of your application executable.	0	matter of resource loading in the application code

Page	Id	Headline	Rule	Res	How
	35	Threads	As you design your application, think about the operations that could be performed in parallel.	1b	the constraints on parallel versus serial (i.e. temporal logic) could be modelled in a transformation which transforms workflows into task models
67	36	Address Book	If your application stores or uses contact information, use the Address Book framework to manage that information.	2b	if address elements are modeled as part of a task diagram it could automatically be attached to the address book dialog by the transformation. Other things must be handeled in code.
	37		You should give the display window a name that describes its contents as they relate to your application, such as "Addresses" or "Contacts."	2b	The transformation could transform task names and the application name into window titles
69	38	Animation	Avoid replicating a Front Row or Time Machine style of user interface for an application that requires a lot of user input (especially textual input) or that users use for content creation.	2c	can not be determined automatically
	39		Avoid animating everything.	2c	can not be determined automatically
	40		Avoid animating routine user-interface actions supported by system-provided controls, such as: Switching tabs, Opening or closing views, Clicking toolbar items.	2a	the transformation will not animate anything but will automatically use system-provided controls
95	41	The Mouse and Other Pointing Devices	Change the cursor's shape only to provide information to the user about changes in the cursor's function.	2b	Tasks / functions can be assigned to specific cursors by the transformation
96	42		Because not everyone is physically able to perform a double click, it should never be the only way to perform an action.	2a	the transformation could transform tasks into at least two trigger methods to perform the task
	43		If the user drags a proxy object to an area that would cause that proxy object to disappear, display the poof cursor to indicate that the proxy object will disappear if dragged to that location.	2b	the transformation could create a cursor model which contains each cursor for each drop operation for a given type of proxy object. In that case poof will be used

Page	Id	Headline	Rule	Res	How
	44		If the user selects an item and begins a drag but releases the item after having moved it three or fewer pixels, the item does not move.	0	handled by the drag and drop animation code
97	45	The Keyboard	Important: Avoid assigning any key combinations listed in the tables in this section to commands other than those specified in the tables. Even if your application doesn't support all the keyboard equivalents shown, don't assign unused combinations to commands that conflict with those specified in this section.	2b	there could be a model element which contains all key combinations associated to tasks. Then it can be transformed into the ui model and assured that these are not the ones listed in the style guide.
	46		Note: Not all the keys described here exist on all keyboards. Don't depend on a key as the only way for users to accomplish a task. You cannot assume anything about which keyboard (if any) is connected to a computer.	2a	the transformation could transform tasks into at least two trigger methods to perform the task, c.f. Id 42
	47		When full keyboard access is turned on, pressing the Space bar selects the item that currently has the keyboard navigation focus (the equivalent of clicking the mouse button).	0	As this is a runtime issue this is implemented in code
98	48		Not all keyboards have a Clear key, so don't require its use in your application.	0	the clear key can only be used in code by attaching appropriate listeners so it is not an isuue of a transformation
	49		Pressing Esc should never cause the user to back out of an operation that would require extensive time or work to reenter. When the user presses Esc during a lengthy operation, display a confirmation dialog to be sure that the key wasn't pressed accidentally.	2b	The tranformation attaches a confirmation window to each task, which could have an attribute lengthy operation which is activated in the code when the user presses the Esc key
	50		You should use these keys Shift, Caps Lock, Option, Command, and Control consistently as described here.	0	as it is subject to the developers mind what is attached to which key this can be modelled but not determined automatically

Page	Id	Headline	Rule	Res	How
99	51		Keyboard combinations using the arrow keys should be used only for shortcuts for mouse actions. It is never appropriate to implement only a keyboard combination and not provide a mouse-based way to perform the same action.	2b	the second half of the guideline is similar to Id 42
100	52		Don't use the arrow keys to: • Move the mouse pointer onscreen • Duplicate the function of the scroll bars	0	must be avoided by the application developer, as most keys are handeled in code
117	53	Drag-and-Drop Overview	Except in cases where drag and drop is so intrinsic to an application that no suitable alternative methods exist dragging icons in the Finder, for example there should always be another method for accomplishing a drag-and-drop task.	2b	the transformation could transform tasks into at least two trigger methods to perform the task, c.f. Id 42
	54		The data that is pasted should be target-specific. For example, if a user drags an Address Book entry to the To text field in Mail, only the email address is pasted, not all of the person's address information.	0	This is application specific and will most likely be handled in code
	55		You should implement Undo for any drag-and-drop operation you enable in your application. If you implement a drag-and-drop operation that is not undoable, display a confirmation dialog before implementing the drop.	2c	it is not possible to generate undos except for reversible languages, so it must be handled in code
	56	Drag-and-Drop Semantics	Your application must determine whether to move or copy a dragged item after it is dropped on a destination.	2b	there could be some pre-defined cases in the transformation which determine whether copy or move is required. However, not all cases can be treated

Page	Id	Headline	Rule	Res	How
119	57	Selection Feedback	Your application should ensure that implicit selection occurs, when appropriate, when the user starts dragging.	0	this is application specific and will most likely be handled in code
	58		When a window containing a highlighted selection becomes inactive, your application should maintain the selection so that users can drag previously selected data from inactive windows to the active window.	0	this should be handled by the toolkit and the code should appropriatly override the focus change event
	59		Items selected only by range-selection operations for example, text or a group of icons must have a background selection to allow the user to drag these items out of inactive windows. Whenever an inactive window is made key, the background selection, if any, becomes highlighted as a normal selection.	0	this should be handled by the toolkit and the code should appropriatly override the focus change event
	60	Drag Feedback	Your application should provide drag feedback as soon as the user drags an item at least three pixels.	2b	the feedback could be generated from given model elements using a transformation. It is the responsibility of the interpreter to activate as given in the guideline
	61		If a user holds the mouse button down on an object or selected text, it should become draggable immediately and stay draggable as long as the mouse remains down.	0	this should be handled by the toolkit and the code should appropriatly override the drag event
	62	Destination Feedback	Destination feedback should not occur simply because your application is drag-aware; rather, it should depend on the destination's ability to accept the type of data contained in the dragged item. For example, a text entry field that accepts only text should not be highlighted when the dragged item is a graphic.	2b	this is part of the result from Id 43

Page	Id	Headline	Rule	Res	How
120	63		Use cursors to indicate what result letting go of the mouse will have. For example, if you are dragging an icon out of a toolbar, show the poof cursor when the user has the icon outside of the toolbar to indicate that if they let go of it there, the item will disappear.	2b	this is part of the result from Id 43
	64		If a drag-and-drop operation takes place entirely within one destination region (moving a document icon to a different location in the same folder window, for example), don't highlight the destination region, to avoid distracting the user. However, if the user drags an item completely out of a destination region and then drags the same item back to the same destination region, the destination region should be highlighted.	2c	This is a runtime issue and cannot be handled in transformations
125	65	Text	Your product development team should include a skilled writer who is responsible for reviewing all user-visible onscreen text as well the instructional documentation. The writer should refer to the Apple Publications Style Guide for guidance on Apple-specific terminology.	2c	can not be assured by a transformation
	66	Fonts	Whenever your application specifies a font, use the system-defined constants shown in Table 10-1 (page 126); avoid using a specific font and point size. Using the system constants ensures that your application always displays the appropriate fonts regardless of changes to the Mac OS.	2b	the fonts could be modelled in a model element and the transformation uses the list whenever a user interface component needs to be paramterized with a font
	67		The system font (Lucida Grande Regular 13 point) is used for text in menus, dialogs, and full-size controls.	2a	those rules could be implemented in the transformation by using only the appropriate font for font parameters for the given components

Page	Id	Headline	Rule	Res	How
	68		Use the emphasized system font (Lucida Grande Bold 13 point) sparingly. It is used for the message text in alerts (see Figure 14-47 (page 233)).	2a	those rules could be implemented in the transformation by using only the appropriate font for font parameters for the given components
	69		The small system font (Lucida Grande Regular 11 point) is used for informative text in alerts (see Figure 14-47 (page 233)). It is also the default font for column headings in lists, for help tags, and for small controls. You can also use it to provide additional information about settings in various windows, such as in the QuickTime preferences.	2a	those rules could be implemented in the transformation by using only the appropriate font for font parameters for the given components
	70		Use the emphasized small system font (Lucida Grande Bold 11 point) sparingly. You might use it to title a group of settings that appear without a group box, or for brief informative text below a text field.	2a	those rules could be implemented in the transformation by using only the appropriate font for font parameters for the given components
	71		The mini system font (Lucida Grande Regular 9 point) is used for mini controls. It can also be used for panel labels and text.	2a	those rules could be implemented in the transformation by using only the appropriate font for font parameters for the given components
	72		An emphasized mini system font (Lucida Grande Bold 9 point) is available for cases in which the emphasized small system font is too large.	1c	it is unknown in advance what is “too large” except that the size can be calculated at design time
	73		If your application creates text documents, use the application font (Lucida Grande Regular 13 point) as the default font for user-created content.	2a	those rules could be implemented in the transformation by using only the appropriate font for font parameters for the given components

Page	Id	Headline	Rule	Res	How
	74		The label font (Lucida Grande Regular 10 point) is used for the labels on toolbar buttons and to label tick marks on full-size sliders. You should rarely need to use this font. For an example of this font used to label a slider control, see the Dock size slider in Dock preferences.	2a	those rules could be implemented in the transformation by using only the appropriate font for font parameters for the given components
	75		Use the view font (Lucida Grande Regular 12 point) as the default font of text in lists and tables.	2a	those rules could be implemented in the transformation by using only the appropriate font for font parameters for the given components
126	76		All user-visible text in your application should be anti-aliased, which is automatic if you use one of the standard system fonts.	2a	since we only apply standard system fonts intransformations
135	77	Icons	To represent your application in Mac OS X, it's essential to create high-quality application icons that scale well in the various places the icon appears the Dock, Finder and Quick Look previews, alert dialogs, and so on.	2c	image processing cannot be done in transformations
136	78	Application Icons	Mac OS X user application icons should be vibrant and inviting, and should immediately convey the application's purpose.	2c	Not machine-readable
137	79		If the primary function of your application is creating or handling media, its icon should display the media the application creates or views. If appropriate, the icon should also contain a tool that communicates the type of task the application allows the user to accomplish. The Preview icon, for example, uses a magnification tool to help convey that the application can be used to view pictures. If you include a supportive tool element, it should closely relate to the base object that it rests upon.	2b	there could be a list of media types pre-defined in the transformation and the transformation associates appropriate icons.

Page	Id	Headline	Rule	Res	How
	80		If you want to "greek" text in an icon, use actual text and make it unreadable by shrinking it or doubling the layers.	2a	doubling text components can be done in a transformation
138	81		Some applications that represent objects or well known products, such as Calculator and QuickTime Player, are most easily recognized by the symbols or objects themselves. When creating icons for such applications, it's more aesthetically pleasing to create a simplified, idealized representation of the object or symbol, instead of using an actual screen shot of the software.	2c	image processing cannot be done in transformations
	82		These icons, many of which are a precursor of what you'll see when you open the application, use a straight-on perspective (rather than the "on a desktop" user application style).	2c	image processing cannot be done in transformations
	83		Because utility applications are normally focused on a narrow set of tasks, it's best to keep the number of elements in the icon to a minimum.	0	Out of scope. Image designer's job
	84		The focus should be a single object that represents what the utility does.	0	Out of scope. Image designer's job
	85	Document Icons	As previously suggested, document icons should make it obvious which application they are associated with. Preview documents, for example, include a graphic of the media (the pictures) used in the application icon.	0	Out of scope. Image designer's job
	86		For simplicity and to avoid confusing the document with the application itself, the viewing tool is not repeated in the document icon.	0	Out of scope. Image designer's job

Page	Id	Headline	Rule	Res	How
139	87		When you want to put an identifying badge over a document icon, treat the badge as an integrated element within the document instead of putting it over the top of the base image and breaking out of the overall document shape.	0	Out of scope. Image designer's job
140	88	Icons for Plug-ins, Hardware, and Removable Media	To help users distinguish between external devices, their icons provide a region for an identifying symbol (FireWire, USB, and so on).	1a	Out of scope. Image designer's job
157	89	Cursors	The standard cursors are designed to provide feedback to users. To maintain a consistent user experience, it is important that you use them only for their intended purpose.	2a	the transformation could create a cursor model which contains each cursor for each task or set of tasks which is aligned with the default cursors C.f. 43
159	90	Standard Cursors	The spinning wait cursor (see Figure 12-1) is displayed automatically by the window server when an application cannot handle all of the events it receives. If an application does not respond for about 2 to 4 seconds, the spinning wait cursor appears. You should try to avoid situations in your application in which the spinning wait cursor will be displayed.	0	this is subject to the code of the application, especially the application logic
	91	Designing Your Own Cursors	Before you design your own cursor, ask yourself if it is going to add value to the user interface. Recognize that by doing so you are introducing a new, potentially confusing user interface element.	0	n/a
161	92	Menus	Ensuring that menus are \emphcorrectly organized, are worded clearly, and behave correctly is crucial to the user's ability to explore and access the functionality of your applications.	2b	there is a general layout which can be implemented. If the application menu items adhere to that standard they can be organized by the transformation. Additionally the names of the tasks of the task model can be used for menu items

Page	Id	Headline	Rule	Res	How
	93		Ensuring that menus are correctly organized, are \emphworded clearly, and behave correctly is crucial to the user's ability to explore and access the functionality of your applications.	2c	not machine-readable
	94		Ensuring that menus are correctly organized, are worded clearly, and \emphbehave correctly is crucial to the user's ability to explore and access the functionality of your applications.	2b	The correct behaviour can be assured by associating the tasks of the task model to the respective menu entries assuming the tasks behave correctly will result in appropriate behaviour
162	95	Menu Behavior	As a general rule, avoid creating long menus. Long menus are difficult for the user to scan and can be overwhelming. If you find that there are too many items in a single menu, try regrouping them; you may find that some of the items fit more naturally in other menus.	2c	as grouping must be done by the developer beacause we don not have a real understanding of what is implemented by the items we cannot automatically regroup the items
	96		If a menu contains more items than are visible onscreen, the menu can scroll to allow the user access to all of the menu items.	0	is handled automatically by the widget toolkit
163	97		Do not design your application to intentionally include scrolling menus; they should exist only when a user adds many items to a customizable menu or when the menu's function causes it to have items added to it (for example, the Finder's Window menu).	2a	they will be used only if a better solution doesn't exist, i.e. not intentionally.
	98	Designing the Elements of Menus	Menu (and submenu) titles should appropriately represent the items in the menu.	2c	Not machine-readable
	99		Make menu titles as short as possible without their losing clarity.	2c	The guideline cannot be applied to us. What is meant here is a design decision, where we generate menus automatically through an algorithm where short menus are prefferred but not guaranteed

Page	Id	Headline	Rule	Res	How
	100		Menu item names should be either actions performed on an object or attributes applied to an object	2c	Not machine-readable
164	101		When a menu item is unavailable because it doesn't apply to the selected object or to the selected object in its current state, or because nothing is selected, for example the item should appear dimmed (gray) in the menu and is not highlighted when the user moves the pointer over it.	0	Out of scope. Runtime issue.
	102		Use title-style capitalization in your menus.	1c	Language processing cannot be done with the transformation language
	103		Dynamic menu items are an appropriate way to offer a shortcut to sophisticated users. However, because dynamic menu items are hidden by default, they should never be the only way to accomplish a task. Be sure that you don't require users to discover a dynamic menu item before they can use your application effectively.	2a	the transformation could transform tasks into at least two trigger methods to perform the task, c.f. Id 42
189	104	Window Appearance	In Mac OS X v10.5 and later, no window-frame surface is visible on the sides of windows;	0	this is the task of the window toolkit
	105		All window-frame areas have a gray gradient surface.	2a	setting colors can be done in the transformation, if color is modelled in the target model
	106		In the window body, content views (such as text or column views) display a white background by default; the surrounding window-body background is a shade of light gray.	2a	setting colors can be done in the transformation, if color is modelled in the target model

Page	Id	Headline	Rule	Res	How
190	107	Window Elements	Every document and application window and panel should have, at a minimum: • A title bar. Even if a window does not have an actual title (a tools panel, for example), it should have a title bar so that users can move the window. • A close button, so that users have a consistent way to dismiss the window.	2b	setting the title can be done in the transformation (using the name of an abstract component), having a close button is subject to the window toolkit
	108		A standard document window may also have the following additional elements that an application window or panel might not have: • Horizontal or vertical scroll bars, or both (if not all the window's contents are visible) • Minimize and zoom buttons • A proxy icon (after a document has been saved) • The title of the document • A resize control • A toolbar control (if the window contains a toolbar)	2b	the scroll bars can be generated by a transformation, the rest is subjec to the window toolkit
191	109		Windows can also display a bottom bar, which is a portion of the window frame that extends below the main content area of the window body.	2b	A transformation can generate content for the bottom bar component and generate a model element representing the bottom bar, depending on if it is needed

Page	Id	Headline	Rule	Res	How
	110		A bottom bar contains controls that directly affect the contents and organization of the window. Controls in a bottom bar are important, but less so than controls in a toolbar.	2a	A transformation can generate content for the bottom bar component and generate a model element representing the bottom bar, depending on if it is needed
192	111		Another optional window element is the scope bar. A scope bar appears below an application's toolbar and allows users to narrow down a search operation or to filter objects or other operations by identifying specific sets of characteristics.	2b	A transformation can generate content for the scope bar component and generate a model element representing the scope bar, depending on if it is needed
193	112		All windows should have a title bar even if the window doesn't have a title (which should be a very rare exception).	2b	setting the title can be done in the transformation (using the name of an abstract component), having a close button is subject to the window toolkit, c.f. Id 107
	113		A document window should display the name of the document being viewed.	2b	setting the title can be done in the transformation (using the name of an abstract component), having a close button is subject to the window toolkit, c.f. Id 107, the name of the document could be implemented using a placeholder in the title, which is added by the transformation
	114		Application windows display the application name.	2a	setting the title can be done in the transformation (using the name of an abstract component), having a close button is subject to the window toolkit, c.f. Id 107
	115		Panels display a descriptive title appropriate for that window.	2a	setting the title can be done in the transformation (using the name of an abstract component), having a close button is subject to the window toolkit, c.f. Id 107

Page	Id	Headline	Rule	Res	How
249	116	Controls	You are strongly encouraged to use standard, system-provided controls in your application.	2a	widgets constructed by these will be packed in a library and used when necessary by the transformation
	117		You should strive to use only the standard control sizes, which are regular, small, and mini.	2a	widgets constructed by these will be packed in a library and used when necessary by the transformation, c.f. 116
	118		In particular, take care to avoid vertically resizing controls. In Mac OS X v10.5 and later, vertically resizing a control can cause it to produce an undesirable look that does not harmonize with the window appearance.	2b	handled by the window toolkit, setting the parameters can be done in a transformation
	119	Window-Frame Controls	A small subset of Mac OS X controls are intended for use only in the window-frame areas, because they have been specially designed to look good on the window-frame surface. These controls should not be used anywhere in the window body. Conversely, all other controls available in Mac OS X can be used in the window body, and most should not be used in the window-frame areas. Of the controls that are designed for use in the window body, there are three that can also be used in window-frame areas: • Icon buttons (including icon buttons that contain a pop-up menu) [...]. • Action menus [...]. • Search fields [...].	2a	these controls can be taken into consideration when creating the widgets in a transformation
250	120		In a toolbar, use rectangular-style toolbar controls to give users access to frequently used commands and objects [...].	0	this is an image design problem and not handled in transformations

Page	Id	Headline	Rule	Res	How
251	121		In a bottom bar, use rectangular-style toolbar controls when you need to provide controls that directly affect the contents or organization of the window body. In general, controls in the bottom bar are important, but they are less frequently used than controls in the toolbar [...].	0	this is an image design problem and not handled in transformations
252	122		A rectangular-style toolbar button can contain either text or icons.	2a	This could be implemented: An abstract component should always have a name (Text) but it could also contain a picture as an alternative representation.
	123		In addition, this button can contain a downward-pointing arrow that indicates the presence of a pop-up menu.	2a	This could be Implemented as another type of target model element representing a new widget (with an arrow)
	124		A rectangular-style toolbar segmented control can also contain either text or icons, but should not contain a mix of text and icons.	2a	the transformation would only produce the one or the other type of model elements representing the widgets not both
	125		If you display an icon in a rectangular toolbar control, be sure the meaning of the image is clear and unambiguous.	0	Out of scope. Image designer's job.
	126		If you want to display text in a rectangular-style toolbar control, be sure it is either a noun (or noun phrase) that describes an object, setting, or state, or that it is a verb (or verb phrase) that describes an action.	2c	Not machine-readable
	127		Also, *if you put such a control in a toolbar, you should provide two descriptive labels that can be displayed below the control.* One label should describe the on (or pressed) state and one should describe the off (or unpressed) state.	2a	The names of the two values of a boolean control can be placed below a boolean component using a transformation

Page	Id	Headline	Rule	Res	How
337	128	Layout Guidelines	As you design the layout of your window, be sure to observe the principle of consistency in the decisions you make [...].	2b	consistency can be partly achieved by always applying the same transformation. However, this is not a guarantee
338	129	Positioning Regular-Size Controls in a Window Body	In Mac OS X, content should always be center-equalized in windows and panes.	0	Exact def. of visual-weight or central-equalized-content missing. Unable to implement with the def..
339	130		When labels and controls are stacked in a group, they should line up with each other vertically.	1a	can be implemented using constraints and a virtual group box container which is not going to be displayed
	131		Its also important to use proper spacing in your window.	1a	spacing can be achieved using constraint solving
341	132		Always center a tab view within a window.	1a	can be implemented using appropriate constraints
342	133		The colons for stacked labels are right-aligned.	1a	can be implemented using appropriate constraints
	134		Stacked controls are left-aligned when appropriate.	1b	can be implemented using appropriate constraints, but when this is appropriate, remains unclear
	135		Similar controls have consistent widths. For example, the sizes of the Font pop-up menus and the Size combo boxes are the same in each group box.	1b	A constraint requiring that components with the same name have equal dimensions could be implemented.
	136		Equal margins between the sides of the group boxes and the tab-pane side edges	1a	can be implemented using appropriate constraints
	137		Equal margins between the sides and bottom of the tab pane and the window edges	1a	can be implemented using appropriate constraints
	138		In both group boxes, the same amount of space between the bottom control and the lower edge of the group box, and the same amount of space between the top control and the upper edge of the group box.	1b	can be implemented using appropriate constraints, escpecially if the space is fixed

Page	Id	Headline	Rule	Res	How
343	139		In addition, the window in Figure 16-8 uses a 12-pixel margin between the top of the tab bar to the bottom of the title bar [. . .]. Note that this margin would be the same width if the window included a toolbar.	1a	can be implemented using appropriate constraints

Table 3: List of Rules and their Results

4 Conclusions

In this technical report we presented the final stage of developement of the Solverational grammar, which is an extension of the QVT Relations language. We printed both, the abstract as well as the concrete syntaxes.

As the biggest novelty of the grammar we introduced optimization into a declarative model-to-model transformation language. Furthermore it is a language which allows for constraint solving. It is a relational language, which - in combination with constraint programming - results ina purely declarative approach.

We showed that it is possible to implement the relations of SGGs using Solverational.

As a result we can conclude that SGGs are faster for certain graphs but Solverational implements all relations of SGGs plus has capabilities for constraint solving and optimization. This makes Solverational more expressive and therefore more transformations can be implemented with Solverational.

Finally we studied a style guide, called the Apple Human Interface Guidelines [1]. We tested how many guidelines can be implemented using Solverational. We found that at least 50% of the rules can be implemented when certain information has been provided with models being transformed.

References

[1] Apple human interface guidelines, 2005.

[2] Jun Kong, Kang Zhang, and Xiaoqin Zeng. Spatial graph grammars for graphical user interfaces. *ACM Trans. Comput.-Hum. Interact.*, 13(2):268–307, 2006.

[3] OMG. Meta object facility 2.0 core final adopted specification. OMG, October 2003.

[4] OMG. Unified modeling language 2.0 infrastructure final adopted specification, September 2003. ptc/03-09-15.

[5] OMG. Object constraint language omg available specification version 2.0. OMG, May 2006.

[6] OMG. Meta object facility (mof) 2.0 query/view/transformation specification. OMG, July 2007. ptc/07-07-07.

[7] Andreas Petter and Alexander Behring. Towards an alignment of declarative modelling and model-to-model transformation languages. In *Informatik 2009*, Lecture Notes in Informatics, Lübeck, Germany, September 2009. GI e.V.

[8] Andreas Petter, Alexander Behring, and Max Mühlhäuser. A methodology for model-driven development of crisis management applications using solverational. In *INFORMATIK 2009*, Lecture Notes in Informatics, Lübeck, Germany, September 2009. GI e.V.

[9] Andreas Petter, Alexander Behring, and Max Mühlhäuser. Solving constraints in model transformations. In Richard F. Paige, editor, *Theory and Practice of Model Transformations*, volume 5563/2009 of *Lecture Notes in Computer Science*, pages 132–147. Springer Berlin / Heidelberg, June 2009.

[10] Andreas Petter, Alexander Behring, Miroslav Zlatkov, Joachim Steinmetz, and Max Mhlhuser. Modeling usability in model-transformations. In Marko Boskovic, Dragan Gasevic, Claus Pahl, and Bernhard Schätz, editors, *Proceedings of the 1st International Workshop on Non-functional System Properties in Domain Specific Modeling Languages, NFPinDSML-2008*, volume 394. CEUR, September 2008. ISSN 1613-0073.

[11] Andreas Petter, Stephan Borgert, Erwin Aitenbichler, Alexander Behring, and Max Mühlhäuser. Optimizing non-functional properties of a service composition using a declarative model-to-model transformation. *Acta Universitates Apulensis*, 18:15, 2009.

[12] Andreas Petter, Stephan Borgert, Erwin Aitenbichler, Alexander Behring, and Max Mühlhäuser. Understanding service composition with non-functional properties using declarative model-to-model transformations. In *Proceedings of the International Symposium on Understanding Intelligent and Complex Systems, UICS 2009*, 2009.

[13] Donna J. Peuquet and Zhan Ci-Xiang. An algorithm to determine the directional relationship between arbitrarily-shaped polygons in the plane. *Pattern Recognition*, 20(1):65 – 74, 1987.