Temporal Change in Product Documentation for Manufacturing

Carsten Sinz and Wolfgang Küchlin
Symbolic Computation Group, WSI for Computer Science,
University of Tübingen, Germany

http://www-sr.informatik.uni-tuebingen.de
Outline of Talk

1\textsuperscript{st} Part: Product documentation for manufacturing at DaimlerChrysler (Mercedes lines):
- Basic documentation schema
- Documentation of temporal aspects

2\textsuperscript{nd} Part: Quality assurance of documentation
- Verification of consistency conditions
- Detection of errors caused by temporal change
- Computation of induced change on parts level
1st Part

Product Documentation for Manufacturing at DaimlerChrysler (Mercedes lines)
Basic Documentation Schema

- Batch configuration for engineering/production
  - Configuration task: order completion, order checking, parts list generation
  - Non-interactive, high throughput system
  - Rule-based, operations controlled by Boolean logic formulae:
    
    \[
    S(x) : \text{supplementing rule for code } x \\
    C(x) : \text{constructability rule for code } x \\
    R(p) : \text{part selection rule for part } p
    \]
  - Atoms: Equipment codes (sales options) and control codes
  - Two levels
    1. function-oriented (equipment codes, product overview)
    2. parts-oriented (modularized hierarchical parts list)
Basic Documentation Schema: Example

Order completion:

\[ S(584) = 906 \vee 955 \vee 625 \land (M104 \lor M112) \]
\[ S(570) = 494 \vee 498 \vee 625 \land 959 \land 955 \]
\[ S(GM) = \neg GA \land (M112 \land L \land M111 \land M104) \]

F202, FW, M111, M23, L, 229L, 744U, 201A, 955

584: electrical window lift
570: shiftable armrest in front
GM: mechanical gearing

08/06/2001 Carsten Sinz, University of Tübingen
Basic Documentation Schema: Example

Constructability check:

\[ C(584) = \neg 583 \]
\[ C(570) = \neg 450 \land \neg 961 \land \neg 962 \land \neg 973 \]
\[ C(GM) = \neg GA \land (M112 \land \neg L \lor M111 \lor M104) \]

F202, FW, M111, M22, L, 229L, 744U, 201A, 955, 584, 570, GM, ...
Basic Documentation Schema: Example

Parts list generation:

\[ R(P49581) = M111 \land \neg 718 \]
\[ R(P49582) = M112 \land 423 \]
\[ R(P49583) = 201A \land 584 \land \neg 715 \]

\[ \vdots \]

F202, FW, M111, M22, L, 229L, 744U, 201A, 955, 584, 570, GM, ...
All rules R equipped with activity time interval:
\[ I(R) = (t_\alpha(R), t_\omega(R)) \]

Additional control codes (CC_\alpha, CC_\omega) may override activity time interval:
- CC_\alpha starts rule activity (even before \( t_\alpha \))
- CC_\omega stops rule activity (even before \( t_\omega \))

[Control codes simplify order-dependent temporal rule selection (e.g. to model overlapping phases).]
Temporal Rules

★ Used in production environment:
  - to control part availability and exchange
  - to model equipment code start-up and run out
  - to accompany assembly line reconfiguration

★ Central question:
How to determine changing part requirements induced by documentation changes?
2nd Part

Quality Assurance of the Product Documentation in the Presence of Temporal Change
Quality Assurance of Documentation

- Static consistency assertions [Küchlin & Sinz, 2000]
  - E.g. no contradictory constructability/supplementing rules, no unused part list entries
  - Boolean formulae, satisfiable if consistency assertion is fulfilled
  - Assertions declare properties holding for all possible constructible orders

- Computation of induced change on parts level
  - Comparison of parts requirement for two scenarios using method for static consistency assertions
Induced Change on Parts Level: Example

<table>
<thead>
<tr>
<th>code/part</th>
<th>rule type</th>
<th>formula</th>
<th>( t_\alpha )</th>
<th>( t_\omega )</th>
</tr>
</thead>
<tbody>
<tr>
<td>P49583</td>
<td>R</td>
<td>201A&amp;584&amp;\neg715</td>
<td>01.04.2001</td>
<td>31.08.2001</td>
</tr>
<tr>
<td>584</td>
<td>C</td>
<td>\neg583</td>
<td>01.01.2001</td>
<td>10.08.2001</td>
</tr>
<tr>
<td>584</td>
<td>C</td>
<td>\bot</td>
<td>11.08.2001</td>
<td>30.09.2001</td>
</tr>
</tbody>
</table>

- Code 584 is valid until 10.08.2001
- Entry P49583 in parts list is unused after 10.08.2001
- Causal chain of dependencies may be arbitrary complex
- Unnecessary parts list entries may also have non-temporal reasons
$\pm \delta$–Method

- Critical change in documentation at time $t_0$, e.g. model year change
- Comparison of parts requirements shortly before and after $t_0$
- Computation of parts requirement at $t_0 \pm \delta$ using techniques from static consistency analysis (SAT)
Comparison of parts requirement at fixed time $t_1$ in the future with and without change

$P_{t_0}$ current parts requirement

$P_{t_1}$ parts requirement without change

$P_{t_1}^*$ parts requirement with change

$P_{t_0}$ documented progress

$P_{t_1}$ intended, yet undocumented change

$P_{t_1}^*$ documented progress and undocumented change

time
Extension: 3-Point Method (continued)

- Intended change is specified only for consistency checking purpose; possible changes:
  - Equipment codes becoming valid or invalid
  - Arbitrary code combinations becoming invalid, e.g. all limousines of model class 202 with 2300cc direct-injection engine (FW∧F202∧M111∧M23)
- Allows handling of production relocation
- Consideration of current parts requirement gives additional information
- Details in the proceedings
Implementation

- Prototypical implementation in C++ as part of our BIS system
- Complete comparison of parts requirements computed in approx. 15-30 min.
- Typical size of input data:
  - 500-1000 Boolean variables (equipment and control codes)
  - 400-750 rules in product overview
  - 4000-10000 rules in parts list
- Promising experiments in parallelization of checks
Summary & Conclusions

- Changes in production/manufacturing are very frequent, thus also in the documentation
- Determination of varying parts requirements is of prime importance
- Conversion to Boolean logic satisfiability problems straightforward
- Application of advanced SAT-checking technology adequate
- Two computation methods for different applications:
  - $\pm\delta$-method for minor changes at fixed time
  - 3-point method for substantial, vaguely specified changes
Thank you for your attention!

Carsten Sinz and Wolfgang Küchlin
Symb. Comp. Group, WSI for Computer Science,
University of Tübingen, Germany

http://www-sr.informatik.uni-tuebingen.de