Masterarbeit

Prüfung intervallbasierter automobiler Konfigurationsdaten

Jakob Breu

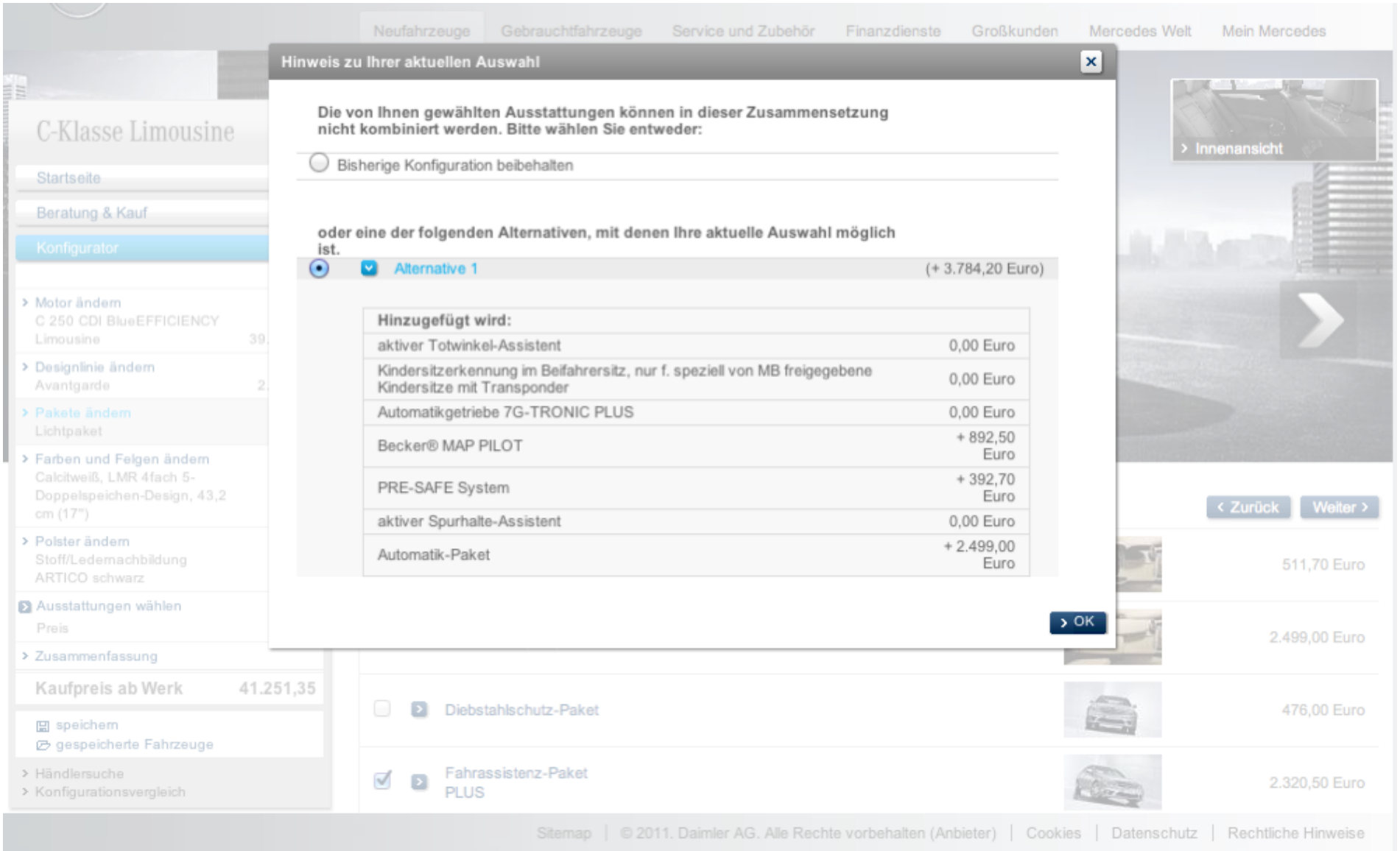
Universität Tübingen

29. September 2011

Agenda

1. Motivation
2. Grundlagen
   - Produktdokumentation
   - Zeitaspekt
3. Konzepte
   - Iterativer Ansatz
   - Explizite Kodierung
   - Abstraction Refinement
4. Implementierung
5. Evaluation
   - Testdaten
   - Erfüllbarkeit
   - Variantentabellen
   - Bewertung
6. Schluss

Motivation



www.mercedes-benz.de - Konfigurator

Motivation

- Premium-Automobilhersteller bieten ihren Kunden immer mehr Wahlmöglichkeiten.

Motivation

- Premium-Automobilhersteller bieten ihren Kunden immer mehr Wahlmöglichkeiten.
- Nicht alle Kombinationen von Einzelteilen sind sinnvoll bzw. physikalisch machbar $\Rightarrow$ Von Mitarbeitern werden Bedingungen formuliert, sogenannte **Regeln** (als Teil der *Produktdokumentation*).

Motivation

- Premium-Automobilhersteller bieten ihren Kunden immer mehr Wahlmöglichkeiten.
- Nicht alle Kombinationen von Einzelteilen sind sinnvoll bzw. physikalisch machbar $\Rightarrow$ Von Mitarbeitern werden Bedingungen formuliert, sogenannte **Regeln** (als Teil der *Produktdokumentation*).
- Die Anzahl dieser Regeln geht inzwischen in die Tausende, und viele Regeln selbst sind kompliziert aufgebaut und haben Abhängigkeiten, die zu beachten sind.

Motivation

- Premium-Automobilhersteller bieten ihren Kunden immer mehr Wahlmöglichkeiten.
- Nicht alle Kombinationen von Einzelteilen sind sinnvoll bzw. physikalisch machbar $\Rightarrow$ Von Mitarbeitern werden Bedingungen formuliert, sogenannte **Regeln** (als Teil der *Produktdokumentation*).
- Die Anzahl dieser Regeln geht inzwischen in die Tausende, und viele Regeln selbst sind kompliziert aufgebaut und haben Abhängigkeiten, die zu beachten sind.
- Mit Hilfe von formalen Prüfmethoden wurde durch Arbeiten des Lehrstuhls „Symbolisches Rechnen“ bei der Daimler AG ein System etabliert, welches für einen Zeitpunkt die Gesamtheit der Regeln auf ihre Korrektheit/Konsistenz prüfen kann:

Motivation

- Premium-Automobilhersteller bieten ihren Kunden immer mehr Wahlmöglichkeiten.
- Nicht alle Kombinationen von Einzelteilen sind sinnvoll bzw. physikalisch machbar $\Rightarrow$ Von Mitarbeitern werden Bedingungen formuliert, sogenannte **Regeln** (als Teil der *Produktdokumentation*).
- Die Anzahl dieser Regeln geht inzwischen in die Tausende, und viele Regeln selbst sind kompliziert aufgebaut und haben Abhängigkeiten, die zu beachten sind.
- Mit Hilfe von formalen Prüfmethoden wurde durch Arbeiten des Lehrstuhls „Symbolisches Rechnen“ bei der Daimler AG ein System etabliert, welches für einen Zeitpunkt die Gesamtheit der Regeln auf ihre Korrektheit/Konsistenz prüfen kann:

Grundlagen

Carsten Sinz: **Verifikation regelbasierter Konfigurationssysteme**, Dissertation, Eberhard Karls Universität Tübingen 2003.

Motivation

- Die Menge der Regeln ist aber nicht statisch: Auslaufende Bauteile, Zulieferer-Wechsel und neue Ausstattungslinien führen zu Gültigkeitsbeschränkungen.

Motivation

- Die Menge der Regeln ist aber nicht statisch: Auslaufende Bauteile, Zulieferer-Wechsel und neue Ausstattungslinien führen zu Gültigkeitsbeschränkungen.
- Je nach Anwendungsbereich (**E**ntwicklung, **P**roduktion oder **A**fter **S**ales) verändert sich die Regelmenge wöchentlich, täglich und manchmal gar stündlich.

Motivation

- Die Menge der Regeln ist aber nicht statisch: Auslaufende Bauteile, Zulieferer-Wechsel und neue Ausstattungslinien führen zu Gültigkeitsbeschränkungen.
- Je nach Anwendungsbereich (**E**ntwicklung, **P**roduktion oder **A**fter **S**ales) verändert sich die Regelmenge wöchentlich, täglich und manchmal gar stündlich.
- Die Betrachtung eines einzelnen Zeitpunktes, ist also nur eine Momentaufnahme, die getroffenen Aussagen sind nicht zwangsläufig länger als ein paar Stunden gültig.

Motivation

- Die Menge der Regeln ist aber nicht statisch: Auslaufende Bauteile, Zulieferer-Wechsel und neue Ausstattungslinien führen zu Gültigkeitsbeschränkungen.
- Je nach Anwendungsbereich (**E**ntwicklung, **P**roduktion oder **A**fter **S**ales) verändert sich die Regelmenge wöchentlich, täglich und manchmal gar stündlich.
- Die Betrachtung eines einzelnen Zeitpunktes, ist also nur eine Momentaufnahme, die getroffenen Aussagen sind nicht zwangsläufig länger als ein paar Stunden gültig.

Motivation

Prüfe die Regelsysteme über ein längeres Zeitintervall, so dass potentielle Probleme vor ihrem Auftreten detektiert werden können $\Rightarrow$ „Prüfung intervallbasierter automobiler Konfigurationsdaten“

Grundlagen

Produktdokumentation

(Voraussetzungen: Aussagenlogik, Erfüllbarkeit, SAT-Solving)

Grundlagen

Produktdokumentation

(Voraussetzungen: Aussagenlogik, Erfüllbarkeit, SAT-Solving)
Code: Ein Code ist eine Variable, die einer Eigenschaft oder einem Bauteil eines Fahrzeugs entspricht.
Bedingung: Eine Bedingung ist eine aussagenlogische Formel.

Grundlagen

Produktdokumentation

(Voraussetzungen: Aussagenlogik, Erfüllbarkeit, SAT-Solving)
Code: Ein Code ist eine Variable, die einer Eigenschaft oder einem Bauteil eines Fahrzeugs entspricht.
Bedingung: Eine Bedingung ist eine aussagenlogische Formel.
Es gibt vier Arten von Regeln:

Grundlagen

Produktdokumentation

(Voraussetzungen: Aussagenlogik, Erfüllbarkeit, SAT-Solving)
Code: Ein Code ist eine Variable, die einer Eigenschaft oder einem Bauteil eines Fahrzeugs entspricht.
Bedingung: Eine Bedingung ist eine aussagenlogische Formel.
Es gibt vier Arten von Regeln:

1. Pauschale Coderegeln

$$c \Rightarrow (b)$$

Die Baubarkeit von c hängt davon ab, ob die Bedingung b erfüllt wird. „b bedingt die Baubarkeit von c.“

Grundlagen

Produktdokumentation

(Voraussetzungen: Aussagenlogik, Erfüllbarkeit, SAT-Solving)
Code: Ein Code ist eine Variable, die einer Eigenschaft oder einem Bauteil eines Fahrzeugs entspricht.
Bedingung: Eine Bedingung ist eine aussagenlogische Formel.
Es gibt vier Arten von Regeln:

1. Pauschale Coderegeln

$$c \Rightarrow (b)$$

Die Baubarkeit von c hängt davon ab, ob die Bedingung b erfüllt wird. „b bedingt die Baubarkeit von c.“

2. Baumusterreferenzen

Die Baumusterreferenzen stellen Alternativen zueinander dar: Sie werden alle disjunktiv aneinander gehängt.

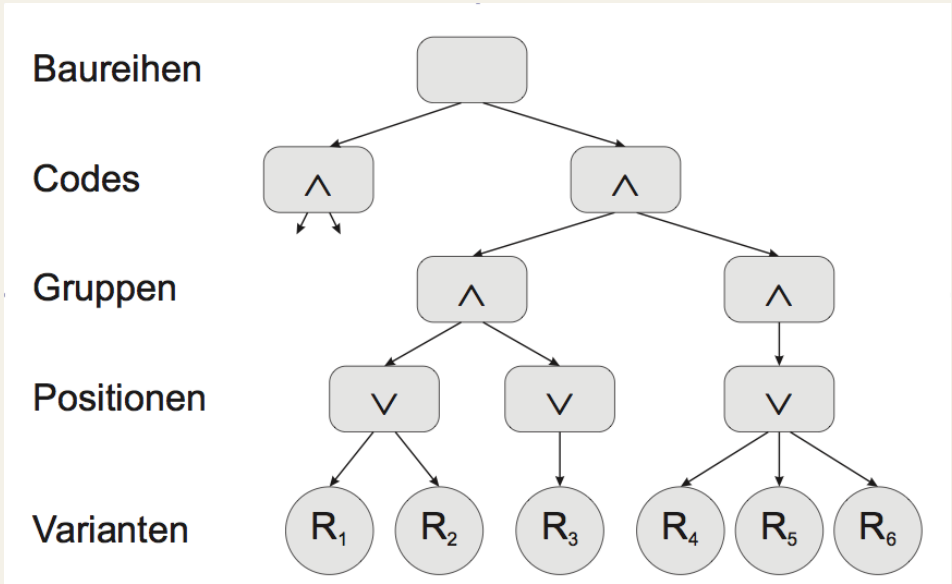
Grundlagen

Produktdokumentation

3. Baubarkeitsregeln

$$c \Rightarrow (b)$$

Ähnlich den pauschalen Coderegeln: Die Baubarkeit von c hängt davon ab, ob die Bedingung b erfüllt wird. „b bedingt die Baubarkeit von c."



Grundlagen

Produktdokumentation

4. Zusteuerungsregeln

$$c \Leftarrow (b)$$

Nicht nur die vom Kunden direkt ausgewählten Teile werden für den Bau eines Fahrzeugs benötigt. Die Zusteuerung sorgt dafür, dass c zu den Eigenschaften oder Bauteilen eines Fahrzeugs hinzugefügt wird, wenn die Bedingung b erfüllt wird. „b steuert c zu.“
Zusteuerungsregeln werden wie Baubarkeitsregeln zu Gruppen, Positionen und Varianten zugeordnet.

Grundlagen

Produktdokumentation

- Für die formale Prüfung werden alle Regeln in eine große Formel zusammengepackt: **Produktübersichtsformel (PÜF)** (Verknüpfung aller Regeln per $\wedge$)

Grundlagen

Produktdokumentation

- Für die formale Prüfung werden alle Regeln in eine große Formel zusammengepackt: **Produktübersichtsformel (PÜF)** (Verknüpfung aller Regeln per $\wedge$)
- Außerdem werden die Baubarkeitsregeln und Zusteuerungsregeln „aggregiert“: Für jeden Code werden die Baubarkeitsregeln sowie die pauschalen Codebedingungen zusammengefasst:

$$c \Rightarrow ((\bigwedge_{a \in P_c} a) \wedge \bigwedge_{g \in G_c} \bigwedge_{p \in POS_{c,g}} \bigvee_{b \in V^B_{c,g,p}} b)$$

Grundlagen

Produktdokumentation

- Für die formale Prüfung werden alle Regeln in eine große Formel zusammengepackt: **Produktübersichtsformel (PÜF)** (Verknüpfung aller Regeln per $\wedge$)
- Außerdem werden die Baubarkeitsregeln und Zusteuerungsregeln „aggregiert“: Für jeden Code werden die Baubarkeitsregeln sowie die pauschalen Codebedingungen zusammengefasst:

$$c \Rightarrow ((\bigwedge_{a \in P_c} a) \wedge \bigwedge_{g \in G_c} \bigwedge_{p \in POS_{c,g}} \bigvee_{b \in V^B_{c,g,p}} b)$$

- Bei den aggregierten Zusteuerungsregeln werden außerdem noch die Baubarkeitsregeln berücksichtigt:

$$c \Leftarrow ((\bigwedge_{a \in P_c} a) \wedge \bigwedge_{g \in G_c} \bigwedge_{p \in POS_{c,g}} \bigvee_{b \in V^B_{c,g,p}, z \in V^Z_{c,g,p}} z \wedge b)$$

Grundlagen

Produktdokumentation

- Für die formale Prüfung werden alle Regeln in eine große Formel zusammengepackt: **Produktübersichtsformel (PÜF)** (Verknüpfung aller Regeln per $\wedge$)
- Außerdem werden die Baubarkeitsregeln und Zusteuerungsregeln „aggregiert“: Für jeden Code werden die Baubarkeitsregeln sowie die pauschalen Codebedingungen zusammengefasst:

$$c \Rightarrow ((\bigwedge_{a \in P_c} a) \wedge \bigwedge_{g \in G_c} \bigwedge_{p \in POS_{c,g}} \bigvee_{b \in V^B_{c,g,p}} b)$$

- Bei den aggregierten Zusteuerungsregeln werden außerdem noch die Baubarkeitsregeln berücksichtigt:

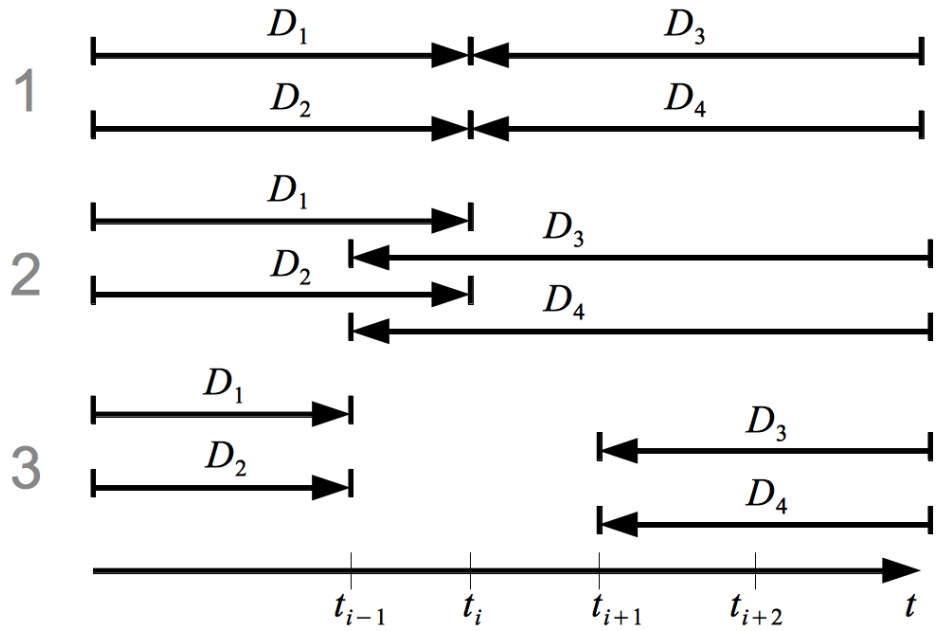
$$c \Leftarrow ((\bigwedge_{a \in P_c} a) \wedge \bigwedge_{g \in G_c} \bigwedge_{p \in POS_{c,g}} \bigvee_{b \in V^B_{c,g,p}, z \in V^Z_{c,g,p}} z \wedge b)$$

- Lenkungsattribute: Regeln können zu links- bzw. rechtsgelenkten Fahrzeugen zugeordnet werden.

Grundlagen

Zeitaspekt

Beispiel: Die Dieselmotoren D_1 und D_2 werden ersetzt durch die neue Generation D_3 und D_4:



Grundlagen

Zeitaspekt

Anwendungsfälle (jeweils bezogen auf ein ausgewähltes Prüfintervall):

- Erfüllbarkeit der Produktübersichtsformeln

Grundlagen

Zeitaspekt

Anwendungsfälle (jeweils bezogen auf ein ausgewähltes Prüfintervall):

- Erfüllbarkeit der Produktübersichtsformeln
- Notwendige Codes

Grundlagen

Zeitaspekt

Anwendungsfälle (jeweils bezogen auf ein ausgewähltes Prüfintervall):

- Erfüllbarkeit der Produktübersichtsformeln
- Notwendige Codes
- Unzulässige Codes

Grundlagen

Zeitaspekt

Anwendungsfälle (jeweils bezogen auf ein ausgewähltes Prüfintervall):

- Erfüllbarkeit der Produktübersichtsformeln
- Notwendige Codes
- Unzulässige Codes
- Prüfung der (segmentierten) Variantentabellen:

Grundlagen

Zeitaspekt

Anwendungsfälle (jeweils bezogen auf ein ausgewähltes Prüfintervall):

- Erfüllbarkeit der Produktübersichtsformeln
- Notwendige Codes
- Unzulässige Codes
- Prüfung der (segmentierten) Variantentabellen:
  - Steuergeräteprogrammierung auf Basis der gewünschten Ausstattungsmerkmale

Grundlagen

Zeitaspekt

Anwendungsfälle (jeweils bezogen auf ein ausgewähltes Prüfintervall):

- Erfüllbarkeit der Produktübersichtsformeln
- Notwendige Codes
- Unzulässige Codes
- Prüfung der (segmentierten) Variantentabellen:
  - Steuergeräteprogrammierung auf Basis der gewünschten Ausstattungsmerkmale
  - Ein **Fragment** besteht aus mehreren **Varianten**

Grundlagen

Zeitaspekt

Anwendungsfälle (jeweils bezogen auf ein ausgewähltes Prüfintervall):

- Erfüllbarkeit der Produktübersichtsformeln
- Notwendige Codes
- Unzulässige Codes
- Prüfung der (segmentierten) Variantentabellen:
  - Steuergeräteprogrammierung auf Basis der gewünschten Ausstattungsmerkmale
  - Ein **Fragment** besteht aus mehreren **Varianten**
  - Die Varianten müssen auf Vollständigkeit, Eindeutigkeit und Erfüllbarkeit geprüft werden

Grundlagen

Zeitaspekt

Anwendungsfälle (jeweils bezogen auf ein ausgewähltes Prüfintervall):

- Erfüllbarkeit der Produktübersichtsformeln
- Notwendige Codes
- Unzulässige Codes
- Prüfung der (segmentierten) Variantentabellen:
  - Steuergeräteprogrammierung auf Basis der gewünschten Ausstattungsmerkmale
  - Ein **Fragment** besteht aus mehreren **Varianten**
  - Die Varianten müssen auf Vollständigkeit, Eindeutigkeit und Erfüllbarkeit geprüft werden

Je nach Anwendungsfall werden zusätzliche **Prüfbedingungen** an die PÜF angehängt.

Konzepte

Konzepte

1. **Iterativer Ansatz:** Für jeden Zeitabschnitt wird eine neue Produktübersichtsformel erzeugt.

Konzepte

1. **Iterativer Ansatz:** Für jeden Zeitabschnitt wird eine neue Produktübersichtsformel erzeugt.
2. **Explizite Kodierung:** Die zeitlichen Veränderungen in den Regeln werden in eine *Gesamt-PÜF* kodiert.

Konzepte

1. **Iterativer Ansatz:** Für jeden Zeitabschnitt wird eine neue Produktübersichtsformel erzeugt.
2. **Explizite Kodierung:** Die zeitlichen Veränderungen in den Regeln werden in eine *Gesamt-PÜF* kodiert.
3. **Abstraction Refinement:** Suche nach einem Zeitabschnitt, in dem die PÜF (und Prüfbedingung) erfüllbar ist.

Konzepte

Iterativer Ansatz

Konzeptionell der einfachste Ansatz: Für jeden Zeitabschnitt wird eine neue Produktübersichtsformel erzeugt.
Vorteile:

- Einfach zu implementieren; wesentliche Module bereits vorhanden (PÜF-Erzeugung, Anwendungsfälle) und in Produktivumgebung eingesetzt.

Konzepte

Iterativer Ansatz

Konzeptionell der einfachste Ansatz: Für jeden Zeitabschnitt wird eine neue Produktübersichtsformel erzeugt.
Vorteile:

- Einfach zu implementieren; wesentliche Module bereits vorhanden (PÜF-Erzeugung, Anwendungsfälle) und in Produktivumgebung eingesetzt.
- Parallelisierbar, jeder Zeitabschnitt kann auf einem separaten System bearbeitet werden.

Konzepte

Iterativer Ansatz

Konzeptionell der einfachste Ansatz: Für jeden Zeitabschnitt wird eine neue Produktübersichtsformel erzeugt.
Vorteile:

- Einfach zu implementieren; wesentliche Module bereits vorhanden (PÜF-Erzeugung, Anwendungsfälle) und in Produktivumgebung eingesetzt.
- Parallelisierbar, jeder Zeitabschnitt kann auf einem separaten System bearbeitet werden.
- Verwendung bereits produktiv eingesetzter Software verspricht Zuverlässigkeit.

Konzepte

Iterativer Ansatz

Konzeptionell der einfachste Ansatz: Für jeden Zeitabschnitt wird eine neue Produktübersichtsformel erzeugt.
Vorteile:

- Einfach zu implementieren; wesentliche Module bereits vorhanden (PÜF-Erzeugung, Anwendungsfälle) und in Produktivumgebung eingesetzt.
- Parallelisierbar, jeder Zeitabschnitt kann auf einem separaten System bearbeitet werden.
- Verwendung bereits produktiv eingesetzter Software verspricht Zuverlässigkeit.
- Optimierungen sind möglich (später mehr).

Nachteil: Für n Zeitabschnitte im Prüfintervall müssen n PÜFs erzeugt (und SAT-Solver-Objekte) werden.

Konzepte

Iterativer Ansatz

Mögliche Optimierungen:

Konzepte

Iterativer Ansatz

Mögliche Optimierungen:

- Wiederverwenden von erfüllenden Belegungen der Variablen. (Implementiert)

Konzepte

Iterativer Ansatz

Mögliche Optimierungen:

- Wiederverwenden von erfüllenden Belegungen der Variablen. (Implementiert)
- Nur ein SAT-Solver-Objekt, in dem die Teile der PÜF, die sich nicht oder selten verändern, „gecachet“ werden. (Nicht implementiert)

Konzepte

Explizite Kodierung

Bei der Expliziten Kodierung werden alle Regeln des Prüfintervalls in eine *Gesamt-PÜF* kodiert. Regeln, die innerhalb dieses Prüfintervalls nicht immer gültig sind, werden mit sog. **Hilfsvariablen** verknüpft, die sie **(de-) aktivieren**.

Konzepte

Explizite Kodierung

Bei der Expliziten Kodierung werden alle Regeln des Prüfintervalls in eine *Gesamt-PÜF* kodiert. Regeln, die innerhalb dieses Prüfintervalls nicht immer gültig sind, werden mit sog. **Hilfsvariablen** verknüpft, die sie **(de-) aktivieren**. Für die Prüfung einzelner Zeitabschnitte werden die Hilfsvariablen entsprechend vorbelegt.

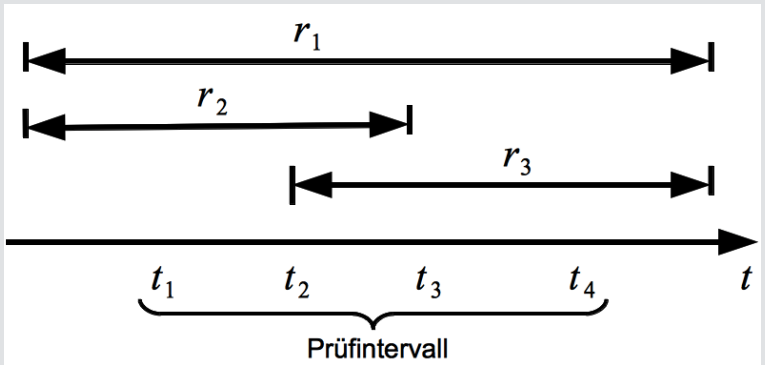
Konzepte

Explizite Kodierung

Bei der Expliziten Kodierung werden alle Regeln des Prüfintervalls in eine *Gesamt-PÜF* kodiert. Regeln, die innerhalb dieses Prüfintervalls nicht immer gültig sind, werden mit sog. **Hilfsvariablen** verknüpft, die sie **(de-) aktivieren**. Für die Prüfung einzelner Zeitabschnitte werden die Hilfsvariablen entsprechend vorbelegt.

Beispiel

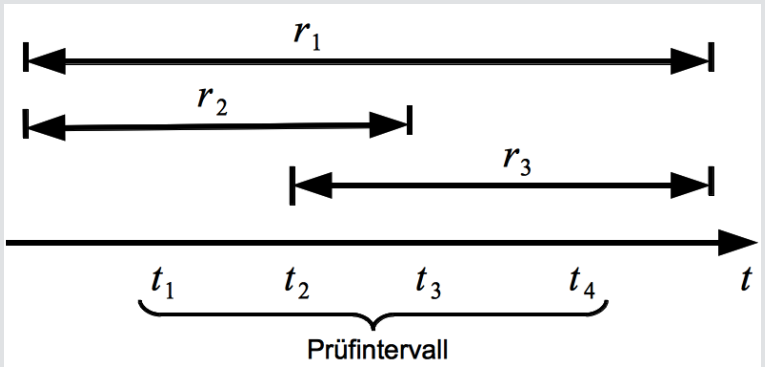
Im Prüfintervall $[t_1, t_4]$ gibt es drei Regeln: r_1 ist immer gültig, r_2 gilt bis t_3, r_3 gilt ab t_2.



Konzepte

Explizite Kodierung

Beispiel

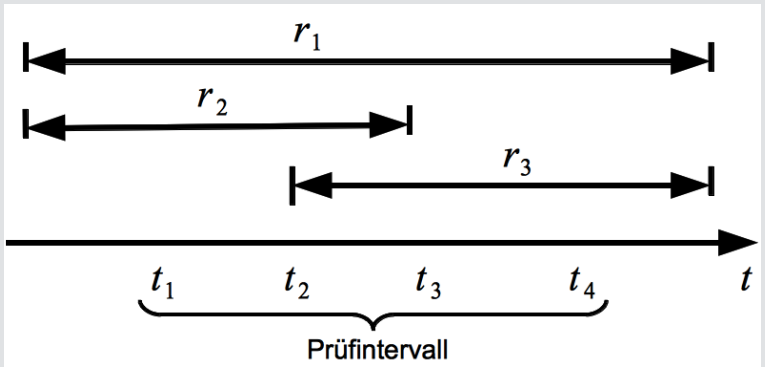


Den Regeln r_2 und r_3 werden die Hilfsvariablen σ_2 bzw. σ_3 zugewiesen:
$r_1 \wedge (\sigma_2 \Rightarrow r_2) \wedge (\sigma_3 \Rightarrow r_3)$

Konzepte

Explizite Kodierung

Beispiel



Den Regeln r_2 und r_3 werden die Hilfsvariablen σ_2 bzw. σ_3 zugewiesen:
$r_1 \wedge (\sigma_2 \Rightarrow r_2) \wedge (\sigma_3 \Rightarrow r_3)$
Ein t_i zur Erfüllbarkeitsprüfung wird nun ausgewählt, in dem eine Vorbelegung der Hilfsvariablen an diesem Zeitabschnitt vorgenommen wird.
t_1: $r_1 \wedge (\sigma_2 \Rightarrow r_2) \wedge (\sigma_3 \Rightarrow r_3) \wedge \sigma_2 \wedge \neg\sigma_3$

Konzepte

Explizite Kodierung

Definition

Die Funktion T ordnet jeder Regel ihr Gültigkeitsintervall zu: $T : r \mapsto (a, b)$

Konzepte

Explizite Kodierung

Definition

Die Funktion T ordnet jeder Regel ihr Gültigkeitsintervall zu: $T : r \mapsto (a, b)$

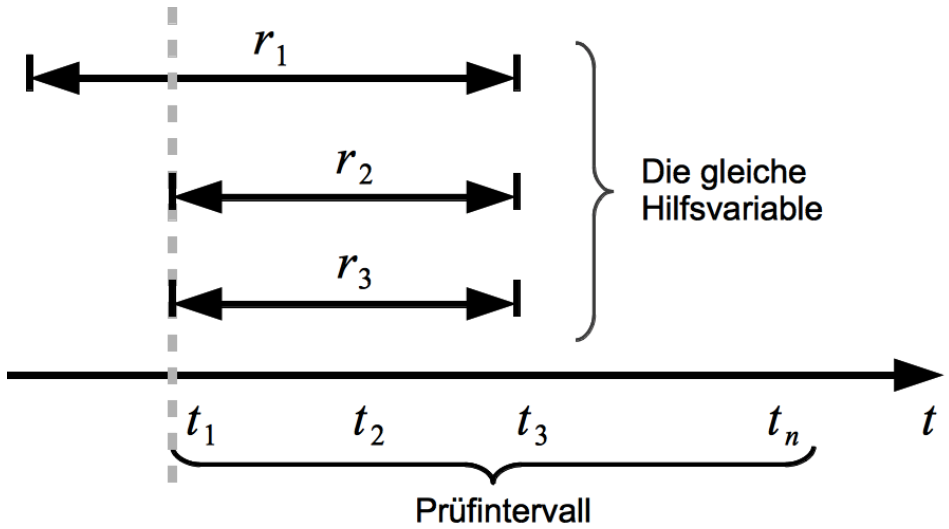
Definition

Die Funktion H ordnet jedem Intervall eine Hilfsvariable zu: $H : (a, b) \mapsto v$
Die Zuordnung genügt dabei folgenden Bedingungen:

1. $T(r) = (a, b), T(r') = (a', b') : a = a' \wedge b = b' \Rightarrow H((a, b)) = H((a', b'))$
   Gleiche Intervalle $\Rightarrow$ gleiche Hilfsvariablen.
2. $T(r) = (a, b), a < s \Rightarrow H((a, b)) = H((s, b))$
   Beginn s $(s \in (D))$ des Prüfintervalls als Beginn aller relevanten Intervalle.
3. $T(r) = (a, b), t < b \Rightarrow H((a, b)) = H((a, t))$
   Analog das Ende t $(t \in (D))$ des Prüfintervalls.

Konzepte

Explizite Kodierung



Konzepte

Explizite Kodierung

Hilfsvariablen-Einsatz nicht immer trivial:

Konzepte

Explizite Kodierung

Hilfsvariablen-Einsatz nicht immer trivial:

- Eine aggregierte Baubarkeits- und Zusteuerungsregel besteht meistens aus mehreren Regeln mit verschiedenen Gültigkeitsdauern: Die Hilfsvariablen werden auf einzelne Varianten angewendet. Sonderfall: Zeitabschnitt mit keiner Variante.

Konzepte

Explizite Kodierung

Hilfsvariablen-Einsatz nicht immer trivial:

- Eine aggregierte Baubarkeits- und Zusteuerungsregel besteht meistens aus mehreren Regeln mit verschiedenen Gültigkeitsdauern: Die Hilfsvariablen werden auf einzelne Varianten angewendet. Sonderfall: Zeitabschnitt mit keiner Variante.
- Ungültige Codes: Codes ohne Baubarkeitsregeln sind ungültig zu machen ($\neg c$); innerhalb eines Prüfintervalls kann ein Code zeitweise gültig und zeitweise ungültig sein.

Konzepte

Explizite Kodierung

Hilfsvariablen-Einsatz nicht immer trivial:

- Eine aggregierte Baubarkeits- und Zusteuerungsregel besteht meistens aus mehreren Regeln mit verschiedenen Gültigkeitsdauern: Die Hilfsvariablen werden auf einzelne Varianten angewendet. Sonderfall: Zeitabschnitt mit keiner Variante.
- Ungültige Codes: Codes ohne Baubarkeitsregeln sind ungültig zu machen ($\neg c$); innerhalb eines Prüfintervalls kann ein Code zeitweise gültig und zeitweise ungültig sein.
- Lenkungsattribute: Bei den aggregierten Baubarkeitsregeln müssen die Vorkommen von Lenkungsattributen berücksichtigt werden. Dieser Fall ist für die Explizite Kodierung schwierig zu implementieren. Zusätzliche *time*-Hilfsvariablen werden eingeführt.

Konzepte

Explizite Kodierung

Hilfsvariablen-Einsatz nicht immer trivial:

- Eine aggregierte Baubarkeits- und Zusteuerungsregel besteht meistens aus mehreren Regeln mit verschiedenen Gültigkeitsdauern: Die Hilfsvariablen werden auf einzelne Varianten angewendet. Sonderfall: Zeitabschnitt mit keiner Variante.
- Ungültige Codes: Codes ohne Baubarkeitsregeln sind ungültig zu machen ($\neg c$); innerhalb eines Prüfintervalls kann ein Code zeitweise gültig und zeitweise ungültig sein.
- Lenkungsattribute: Bei den aggregierten Baubarkeitsregeln müssen die Vorkommen von Lenkungsattributen berücksichtigt werden. Dieser Fall ist für die Explizite Kodierung schwierig zu implementieren. Zusätzliche *time*-Hilfsvariablen werden eingeführt.

Optimierung: Andere Systematik für die Verwendung/Zuordnung der Hilfsvariablen.

Konzepte

Abstraction Refinement

Ziel dieses Ansatz: Finde einen Zeitabschnitt, an dem die PÜF (und Prüfbedingung) erfüllbar ist.

Prinzip des Abstraction Refinement

Für ein abstraktes Modell wird eine Lösung gesucht (Abstraction), die die Bedingungen des konkreten Modells erfüllt. Wenn diese Lösung äußeren Vorgaben entspricht, kann sie ausgegeben werden.

Konzepte

Abstraction Refinement

Ziel dieses Ansatz: Finde einen Zeitabschnitt, an dem die PÜF (und Prüfbedingung) erfüllbar ist.

Prinzip des Abstraction Refinement

Für ein abstraktes Modell wird eine Lösung gesucht (Abstraction), die die Bedingungen des konkreten Modells erfüllt. Wenn diese Lösung äußeren Vorgaben entspricht, kann sie ausgegeben werden.
Ist dies nicht der Fall, wird das Modell um die Negation dieser Lösung ergänzt (Refinement), so dass die nächste gefundene Lösung nicht der vorherigen gleichen kann.

Konzepte

Abstraction Refinement

Ziel dieses Ansatz: Finde einen Zeitabschnitt, an dem die PÜF (und Prüfbedingung) erfüllbar ist.

Prinzip des Abstraction Refinement

Für ein abstraktes Modell wird eine Lösung gesucht (Abstraction), die die Bedingungen des konkreten Modells erfüllt. Wenn diese Lösung äußeren Vorgaben entspricht, kann sie ausgegeben werden.
Ist dies nicht der Fall, wird das Modell um die Negation dieser Lösung ergänzt (Refinement), so dass die nächste gefundene Lösung nicht der vorherigen gleichen kann.

„Durchprobieren von nicht gewünschten Lösungen, bis die Richtige gefunden wurde.“

Konzepte

Abstraction Refinement

Funktionsweise:

1. Erzeuge die Gesamt-PÜF über das Prüfintervall (analog Explizite Kodierung).

Konzepte

Abstraction Refinement

Funktionsweise:

1. Erzeuge die Gesamt-PÜF über das Prüfintervall (analog Explizite Kodierung).
2. Speichere zu den Zeitabschnitten im Prüfintervall die Belegungen der Hilfsvariablen ab.

Konzepte

Abstraction Refinement

Funktionsweise:

1. Erzeuge die Gesamt-PÜF über das Prüfintervall (analog Explizite Kodierung).
2. Speichere zu den Zeitabschnitten im Prüfintervall die Belegungen der Hilfsvariablen ab.
3. Führe eine Erfüllbarkeitsprüfung der Gesamt-PÜF durch.

Konzepte

Abstraction Refinement

Funktionsweise:

1. Erzeuge die Gesamt-PÜF über das Prüfintervall (analog Explizite Kodierung).
2. Speichere zu den Zeitabschnitten im Prüfintervall die Belegungen der Hilfsvariablen ab.
3. Führe eine Erfüllbarkeitsprüfung der Gesamt-PÜF durch.
4. Ist sie unerfüllbar, gibt es keinen Zeitabschnitt, an dem sie erfüllbar ist $\rightarrow$ Abbruch.

Konzepte

Abstraction Refinement

Funktionsweise:

1. Erzeuge die Gesamt-PÜF über das Prüfintervall (analog Explizite Kodierung).
2. Speichere zu den Zeitabschnitten im Prüfintervall die Belegungen der Hilfsvariablen ab.
3. Führe eine Erfüllbarkeitsprüfung der Gesamt-PÜF durch.
4. Ist sie unerfüllbar, gibt es keinen Zeitabschnitt, an dem sie erfüllbar ist $\rightarrow$ Abbruch.
5. Andernfalls: Filtere aus der erfüllenden Belegung die Hilfsvariablen heraus und schaue, ob ihre Belegung einem Zeitabschnitt t im Prüfintervall entspricht.

Konzepte

Abstraction Refinement

Funktionsweise:

1. Erzeuge die Gesamt-PÜF über das Prüfintervall (analog Explizite Kodierung).
2. Speichere zu den Zeitabschnitten im Prüfintervall die Belegungen der Hilfsvariablen ab.
3. Führe eine Erfüllbarkeitsprüfung der Gesamt-PÜF durch.
4. Ist sie unerfüllbar, gibt es keinen Zeitabschnitt, an dem sie erfüllbar ist $\rightarrow$ Abbruch.
5. Andernfalls: Filtere aus der erfüllenden Belegung die Hilfsvariablen heraus und schaue, ob ihre Belegung einem Zeitabschnitt t im Prüfintervall entspricht.
6. Falls ja: Zum Zeitpunkt t existiert eine Lösung $\rightarrow$ Fertig.

Konzepte

Abstraction Refinement

Funktionsweise:

1. Erzeuge die Gesamt-PÜF über das Prüfintervall (analog Explizite Kodierung).
2. Speichere zu den Zeitabschnitten im Prüfintervall die Belegungen der Hilfsvariablen ab.
3. Führe eine Erfüllbarkeitsprüfung der Gesamt-PÜF durch.
4. Ist sie unerfüllbar, gibt es keinen Zeitabschnitt, an dem sie erfüllbar ist $\rightarrow$ Abbruch.
5. Andernfalls: Filtere aus der erfüllenden Belegung die Hilfsvariablen heraus und schaue, ob ihre Belegung einem Zeitabschnitt t im Prüfintervall entspricht.
6. Falls ja: Zum Zeitpunkt t existiert eine Lösung $\rightarrow$ Fertig.
7. Andernfalls: Negiere die Konjunktion der Belegung der Hilfsvariablen und füge sie an die PÜF an.

Konzepte

Abstraction Refinement

Funktionsweise:

1. Erzeuge die Gesamt-PÜF über das Prüfintervall (analog Explizite Kodierung).
2. Speichere zu den Zeitabschnitten im Prüfintervall die Belegungen der Hilfsvariablen ab.
3. Führe eine Erfüllbarkeitsprüfung der Gesamt-PÜF durch.
4. Ist sie unerfüllbar, gibt es keinen Zeitabschnitt, an dem sie erfüllbar ist $\rightarrow$ Abbruch.
5. Andernfalls: Filtere aus der erfüllenden Belegung die Hilfsvariablen heraus und schaue, ob ihre Belegung einem Zeitabschnitt t im Prüfintervall entspricht.
6. Falls ja: Zum Zeitpunkt t existiert eine Lösung $\rightarrow$ Fertig.
7. Andernfalls: Negiere die Konjunktion der Belegung der Hilfsvariablen und füge sie an die PÜF an.
8. Gehe zu Schritt 3.

Konzepte

Abstraction Refinement

Optimierung: Abstraction Refinement mit UNSAT Cores Sukzessives Entfernen

von Zeitabschnitten, für die keine erfüllende Belegung existieren kann.

Implementierung

Programmierung aller drei Ansätze und zweier Anwendungsfälle (Erfüllbarkeit und Variantentabellen) mit Java.

Implementierung

Programmierung aller drei Ansätze und zweier Anwendungsfälle (Erfüllbarkeit und Variantentabellen) mit Java.
Vorgaben:

Implementierung

Programmierung aller drei Ansätze und zweier Anwendungsfälle (Erfüllbarkeit und Variantentabellen) mit Java.
Vorgaben:

- Auswahl des Prüfintervalls (unrelevante Zeitabschnitte ignorieren)

Implementierung

Programmierung aller drei Ansätze und zweier Anwendungsfälle (Erfüllbarkeit und Variantentabellen) mit Java.
Vorgaben:

- Auswahl des Prüfintervalls (unrelevante Zeitabschnitte ignorieren)
- Vermeidung von algorithmischer Komplexität

Implementierung

Programmierung aller drei Ansätze und zweier Anwendungsfälle (Erfüllbarkeit und Variantentabellen) mit Java.
Vorgaben:

- Auswahl des Prüfintervalls (unrelevante Zeitabschnitte ignorieren)
- Vermeidung von algorithmischer Komplexität
- Codedokumentation

Implementierung

Programmierung aller drei Ansätze und zweier Anwendungsfälle (Erfüllbarkeit und Variantentabellen) mit Java.
Vorgaben:

- Auswahl des Prüfintervalls (unrelevante Zeitabschnitte ignorieren)
- Vermeidung von algorithmischer Komplexität
- Codedokumentation
- Einsatz objektorientierter Paradigmen

Evaluation

Testdaten

Evaluation

Testdaten

Regeln in vier Tabellen (je Typ eine):

	Dateiname	**#Regeln**	**#Zeiten (P)**	**#Zeiten (K)**
Gesamt		16502	227	2083
Pauschale Code-bedingungen	C204PCDP.csv	9243	110	1073
Baubarkeitsregeln	C204X4EP.csv	4902	93	976
Zusteuerungsregeln	C204X4EZ.csv	2236	79	672
Baumusterreferenzen	C204X2E.csv	121		55

Evaluation

Testdaten

Regeln in vier Tabellen (je Typ eine):

	Dateiname	**#Regeln**	**#Zeiten (P)**	**#Zeiten (K)**
Gesamt		16502	227	2083
Pauschale Code-bedingungen	C204PCDP.csv	9243	110	1073
Baubarkeitsregeln	C204X4EP.csv	4902	93	976
Zusteuerungsregeln	C204X4EZ.csv	2236	79	672
Baumusterreferenzen	C204X2E.csv	121		55

Anzahl der Codes, Gruppen, Positionen und (Positions-) Varianten:

Codes	2485
Gruppen	16
Positionen	1261
Varianten	274

Evaluation

Testdaten

Regeln in vier Tabellen (je Typ eine):

	Dateiname	**#Regeln**	**#Zeiten (P)**	**#Zeiten (K)**
Gesamt		16502	227	2083
Pauschale Code-bedingungen	C204PCDP.csv	9243	110	1073
Baubarkeitsregeln	C204X4EP.csv	4902	93	976
Zusteuerungsregeln	C204X4EZ.csv	2236	79	672
Baumusterreferenzen	C204X2E.csv	121		55

Anzahl der Codes, Gruppen, Positionen und (Positions-) Varianten:

Codes	2485
Gruppen	16
Positionen	1261
Varianten	274

In den Variantentabellen 742 Fragmente, auf die sich 1982 Varianten verteilen.

Ausstattungsvariante „FW“ (Limousine) der aktuellen C-Klasse.

Evaluation

Erfüllbarkeit

```
Vorbereitung... abgeschlossen!
Ergebnisse der Prüfung auf Erfüllbarkeit:
Methode: Iterative Prüfung
Prüfintervall: Thu Jan 01 00:00:00 CET 2009 bis Sat Jan 01 00:00:00 CET 2011
Zeitabschnitte im Intervall: 63
Dauer der Prüfung: 46380ms (Vorbereitung 44314, Schleife 2066)
Anzahl der Prüfungen: 63
 davon SAT: 63 (100.0%)
 davon UNSAT: 0 (0.0%)
```

```
Vorbereitung... abgeschlossen!
Ergebnisse der Prüfung auf Erfüllbarkeit:
Methode: Explizite Kodierung
Prüfintervall: Thu Jan 01 00:00:00 CET 2009 bis Sat Jan 01 00:00:00 CET 2011
Zeitabschnitte im Intervall: 63
Dauer der Prüfung: 8371ms (Vorbereitung 4948, Schleife 3423)
Anzahl der Prüfungen: 63
 davon SAT: 63 (100.0%)
 davon UNSAT: 0 (0.0%)
```

Evaluation

Erfüllbarkeit

```
Vorbereitung... abgeschlossen!
Ergebnisse der Prüfung auf Erfüllbarkeit:
Methode: Abstraction Refinement
Prüfintervall: Sun Aug 01 00:00:00 CEST 2010 bis Sat Jan 01 00:00:00 CET 2011
Zeitabschnitte im Intervall: 12
Es wurde ein Zeitpunkt im Prüfintervall gefunden, an dem die PÜF erfüllbar ist.
Dauer der Prüfung: 16815ms (Vorbereitung 4333, Schleife 12482)
```

Exponentiell in der Anzahl der Hilfsvariablen steigende Laufzeit.

Evaluation

Variantentabellen

```
Ergebnisse der Prüfung auf Vollständigkeit, Mehrdeutigkeit und Erfüllbarkeit:
Prüfintervall: Thu Jan 01 00:00:00 CET 2009 bis Fri Jan 01 00:00:00 CET 2010
Zeitabschnitte im Intervall: 29
Anzahl der geprüften Fragmente: 736
Anzahl der Fragmente, die im Prüfintervall vollständig, eindeutig und erfüllbar
sind: 630
Anzahl der Fragmente, bei denen das nicht der Fall ist: 106

Methode: Iterative Prüfung
Längste PÜF: 351800 Zeichen
Dauer der Prüfung: 349938ms (Vorbereitung 36993, Schleife 312945)

Methode: Explizite Kodierung
Anzahl der Hilfsvariablen: 47
Länge der Gesamt-PÜF: 355728 Zeichen
Dauer der Prüfung: 294024ms (Vorbereitung 6416, Schleife 287608)

[...]
Fragment 620544: eeeeeeeeeeeeeeeeeeeeeeeeeeeeee
Fragment 512300: mmmmmmmmmmmmmmmmmmmmmmmmmmmmm
Fragment 630441: eeeeeeeeeeeeeeeeeeeeeeeeeeeeee
Fragment 630442: ----------------------------e
[...]
```

Evaluation

Bewertung

- Die Laufzeiten von iterativem Ansatz und Expliziter Kodierung sind besser als erwartet. Ein Jahr lässt sich in ca. 5-6 Minuten prüfen.

Evaluation

Bewertung

- Die Laufzeiten von iterativem Ansatz und Expliziter Kodierung sind besser als erwartet. Ein Jahr lässt sich in ca. 5-6 Minuten prüfen.
- Ebenso stimmen die Ergebnisse **meist** überein.

Evaluation

Bewertung

- Die Laufzeiten von iterativem Ansatz und Expliziter Kodierung sind besser als erwartet. Ein Jahr lässt sich in ca. 5-6 Minuten prüfen.
- Ebenso stimmen die Ergebnisse **meist** überein.
- Unterschiede treten in einigen Fällen bei den „Zeitstrahlen“ der Variantentabellen auf. Für den Zeitraum 1.1.2010 bis 1.1.2011 werden 127 (iterativ) und 123 (explizit) fehlerhafte Fragmente gemeldet.

Evaluation

Bewertung

- Die Laufzeiten von iterativem Ansatz und Expliziter Kodierung sind besser als erwartet. Ein Jahr lässt sich in ca. 5-6 Minuten prüfen.
- Ebenso stimmen die Ergebnisse **meist** überein.
- Unterschiede treten in einigen Fällen bei den „Zeitstrahlen“ der Variantentabellen auf. Für den Zeitraum 1.1.2010 bis 1.1.2011 werden 127 (iterativ) und 123 (explizit) fehlerhafte Fragmente gemeldet.
- Der iterative Ansatz wird dabei als „Benchmark“ aufgefasst $\Rightarrow$ Unzulänglichkeiten bei der Implementierung der Expliziten Kodierung: Die Sonderfälle bei Positionsvarianten, Lenkungsattributen und unzulässigen Codes müssen umsichtiger programmiert werden.

Evaluation

Bewertung

- Die Laufzeiten von iterativem Ansatz und Expliziter Kodierung sind besser als erwartet. Ein Jahr lässt sich in ca. 5-6 Minuten prüfen.
- Ebenso stimmen die Ergebnisse **meist** überein.
- Unterschiede treten in einigen Fällen bei den „Zeitstrahlen“ der Variantentabellen auf. Für den Zeitraum 1.1.2010 bis 1.1.2011 werden 127 (iterativ) und 123 (explizit) fehlerhafte Fragmente gemeldet.
- Der iterative Ansatz wird dabei als „Benchmark“ aufgefasst $\Rightarrow$ Unzulänglichkeiten bei der Implementierung der Expliziten Kodierung: Die Sonderfälle bei Positionsvarianten, Lenkungsattributen und unzulässigen Codes müssen umsichtiger programmiert werden.
- Abstraction Refinement in der implementierten Version nur eingeschränkt einsetzbar.

Fazit

- Es besteht ein Bedarf für die intervallbasierte Konfigurationsdatenprüfung.

Fazit

- Es besteht ein Bedarf für die intervallbasierte Konfigurationsdatenprüfung.
- Der iterative Ansatz verspricht eine zügige Implementierung für den Produktiveinsatz.

Fazit

- Es besteht ein Bedarf für die intervallbasierte Konfigurationsdatenprüfung.
- Der iterative Ansatz verspricht eine zügige Implementierung für den Produktiveinsatz.
- Die Explizite Kodierung ermöglicht schnellere Prüfungen, ist aber diffizil in der Implementierung.

Fazit

- Es besteht ein Bedarf für die intervallbasierte Konfigurationsdatenprüfung.
- Der iterative Ansatz verspricht eine zügige Implementierung für den Produktiveinsatz.
- Die Explizite Kodierung ermöglicht schnellere Prüfungen, ist aber diffizil in der Implementierung.
- Das Abstraction Refinement muss beschleunigt werden, um mitzuhalten.

Fazit

- Es besteht ein Bedarf für die intervallbasierte Konfigurationsdatenprüfung.
- Der iterative Ansatz verspricht eine zügige Implementierung für den Produktiveinsatz.
- Die Explizite Kodierung ermöglicht schnellere Prüfungen, ist aber diffizil in der Implementierung.
- Das Abstraction Refinement muss beschleunigt werden, um mitzuhalten.
- Für alle drei Konzepte gibt es Optimierungsmöglichkeiten.

Fazit

- Es besteht ein Bedarf für die intervallbasierte Konfigurationsdatenprüfung.
- Der iterative Ansatz verspricht eine zügige Implementierung für den Produktiveinsatz.
- Die Explizite Kodierung ermöglicht schnellere Prüfungen, ist aber diffizil in der Implementierung.
- Das Abstraction Refinement muss beschleunigt werden, um mitzuhalten.
- Für alle drei Konzepte gibt es Optimierungsmöglichkeiten.
- Eine intervallbasierte Konfigurationsdatenprüfung ist mit akzeptablen Laufzeiten möglich.

Fazit

- Es besteht ein Bedarf für die intervallbasierte Konfigurationsdatenprüfung.
- Der iterative Ansatz verspricht eine zügige Implementierung für den Produktiveinsatz.
- Die Explizite Kodierung ermöglicht schnellere Prüfungen, ist aber diffizil in der Implementierung.
- Das Abstraction Refinement muss beschleunigt werden, um mitzuhalten.
- Für alle drei Konzepte gibt es Optimierungsmöglichkeiten.
- Eine intervallbasierte Konfigurationsdatenprüfung ist mit akzeptablen Laufzeiten möglich.
- Eine Übertragung in den Produktiveinsatz kann damit begonnen werden.

Vielen Dank für die Aufmerksamkeit!