Practical Applications of SAT

Carsten Sinz

Symbolic Computation Group, WSI

University of Tübingen

Germany

Overview

- Well-known applications
- Industrial case studies:
  1. Logic-based configuration (DaimlerChrysler)
  2. Rule-based expert system (IBM)
- Implications on logical formalisms, practical experiences
- Complexity considerations

Well-Known Applications

Well-Known Applications of SAT (I)

- Bounded Model Checking (BMC)
  - Introduced by Biere, Clarke et al., '99.
  - Model checking problem:

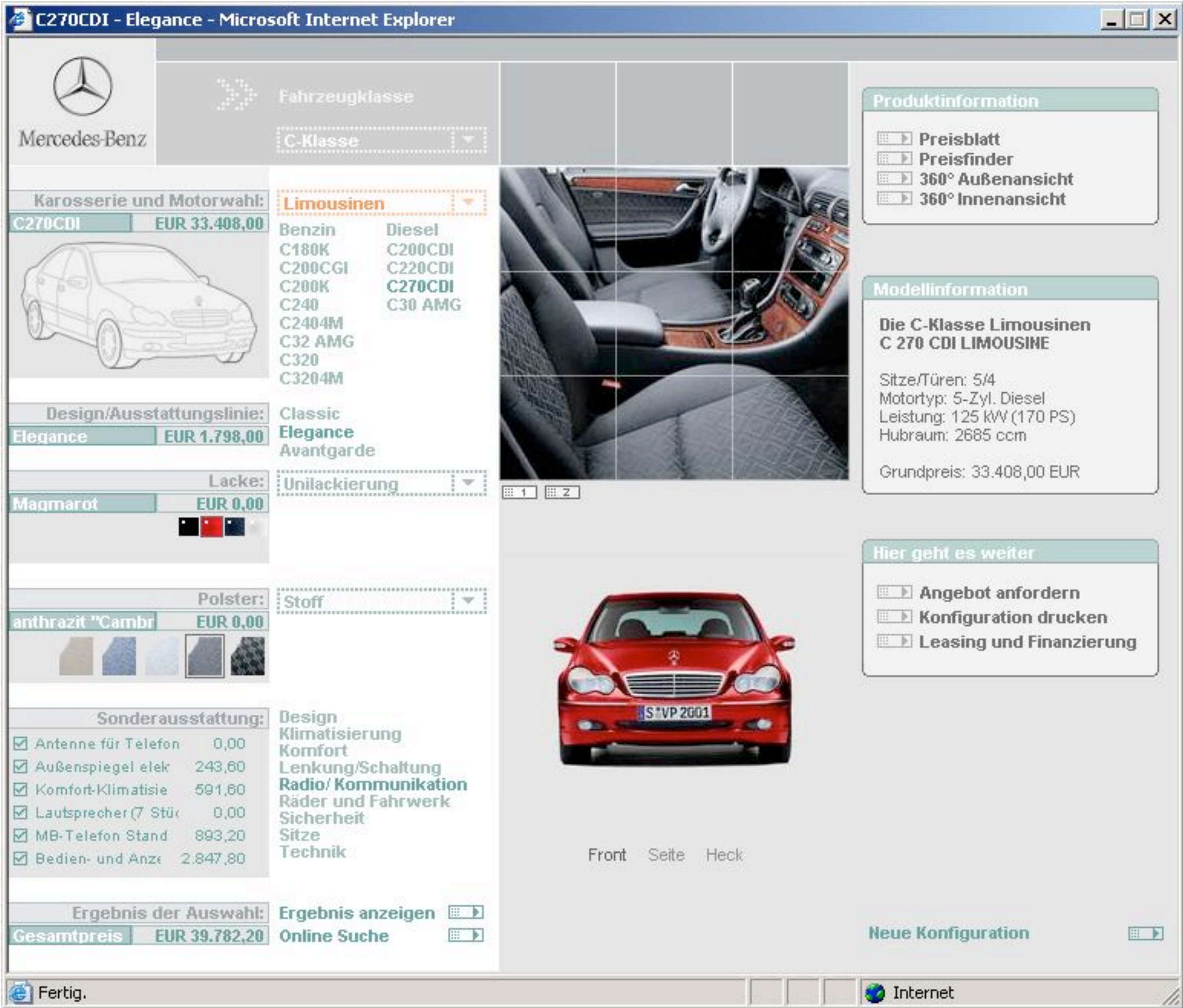
    Input: Finite automaton A (given as transition relation R) and property P (in some temporal logic)

    Question: Does P hold in A? (on all paths up to length *k*)
  - Concrete examples:
    - Cache Coherence of Alpha Microprocessor at Compaq
    - Verification of Sun PicoJava II microprocessor [McMillan]
    - Hardware verification at IBM Haifa (using RuleBase)
  - Formulae of up to $\approx 10^6$ variables, "structured"
  - BMC used for "debugging" hardware designs

Well-Known Applications of SAT (II)

- Planning [Kautz,McAllester,Selman '96]
  - STRIPS-planning: set A of actions; for each $a \in A$: precondition P_a and effect E_a of a (propositional formulae)
  - Encoding schema: $(a(t) \Rightarrow P_a(\mathrm{t})) \wedge (a(t) \Rightarrow E_a(t+1))$ for times t.
  - Problem: Find action sequence $(a(t))_{t \leq T}$ such that goal property $G(t)$ holds at time T.
- Cryptanalysis and finite mathematics
  - DES: Given a set of plaintext / ciphertext blocks, what is the encryption key? [Massacci,Marraro 2000]
  - Latin square completion [e.g. Gomes et al.]
  - Quasigroup existence [H. Zhang '94]
- and others: scheduling, error correcting codes,…

Industrial Application I:
Automotive Product Configuration

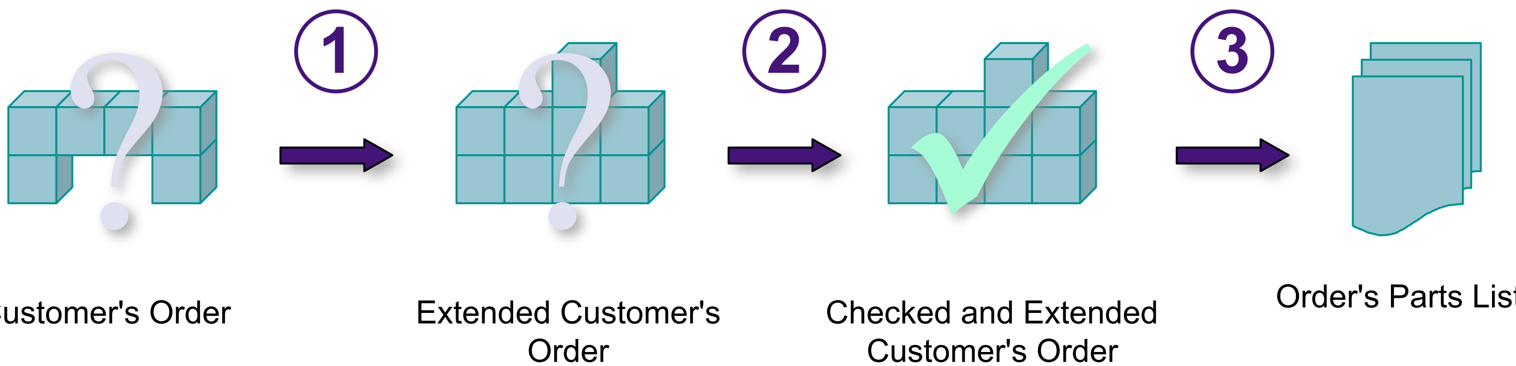

C270CDI - Elegance - Microsoft Internet Explorer
Mercedes-Benz
Fahrzeugklasse
C-Klasse
Karosserie und Motorwahl:
C270CDI
EUR 33.408,00
Limousinen
Benzin
C180K
C200CGI
C200K
C240
C2404M
C32 AMG
C320
C3204M
Diesel
C200CDI
C220CDI
C270CDI
C30 AMG
Design/Ausstattungslinie:
Elegance
EUR 1.798,00
Classic
Elegance
Avantgarde
Lacke:
Magmarot
EUR 0,00
Unilackierung
Polster:
anthrazit "Cambr
EUR 0,00
Stoff
Sonderausstattung:
Antenne für Telefon 0,00
Außenspiegel elek 243,60
Komfort-Klimatisie 591,60
Lautsprecher (7 Stü 0,00
MB-Telefon Stand 893,20
Bedien- und Anze 2.847,80
Design
Klimatisierung
Komfort
Lenkung/Schaltung
Radio/ Kommunikation
Räder und Fahrwerk
Sicherheit
Sitze
Technik
Ergebnis der Auswahl:
Gesamtpreis
EUR 39.782,20
Ergebnis anzeigen
Online Suche
Front
Seite
Heck
Produktinformation
Preisblatt
Preisfinder
360° Außenansicht
360° Innenansicht
Modellinformation
Die C-Klasse Limousinen
C 270 CDI LIMOUSINE
Sitze/Türen: 5/4
Motortyp: 5-Zyl. Diesel
Leistung: 125 kW (170 PS)
Hubraum: 2685 ccm
Grundpreis: 33.408,00 EUR
Hier geht es weiter
Angebot anfordern
Konfiguration drucken
Leasing und Finanzierung
Neue Konfiguration
Fertig.
Internet

Automotive Product Configuration (I)

- Scenario:
  - Electronic configuration of Mercedes car and truck lines
  - Rule-based EPDM (Electronic Product Data Management) system already present
  - Boolean logic employed to express constraints and to control processing of orders
- Problem:
  - Complexity of product and documentation induces errors
- Goals:
  - Computer-based assistance in finding potential errors
  - Increasing documentation quality

Automotive Product Configuration (II)

- Automatic order processing in three steps:
  1. Order completion
  2. Constructibility check
  3. Parts list generation
- All steps controlled by evaluating logical rules



Automotive Product Configuration (III)

- Rules check and modify orders, generate parts-list:

  682 ← 513L ∨ 727L add equipment for fire extinguisher (682) if car goes to Belgium (513L) or Guatemala (727L)

  970 → 673 ∧ 260 all police cars (970) must be equipped with a high-capacity battery (673) and no model type indicator on boot (260)

  Z04 ∨ Z06 ⇢ P9476 add special sealing of driver's door (P9476) to parts-list if car is armored (of type Z04 and Z06)

- Up to approx. 1,900 variables and 10,000 rules
- Consistency of rule system? Implications of change?

Automotive Product Configuration (IV)

- Formalization of order processing algorithm *P* in PDL (propositional dynamic logic):

  do [order completion]

  $Z_1 \wedge \neg x_1 \rightarrow x_1$:=true |…|

  $Z_n \wedge \neg x_n \rightarrow x_n$:=true

  od

  for i=1 to n do [constructibility check]

  if $x_i \wedge \neg B_i$ then fail

  … [parts list generation]

- Typical consistency test: $\langle P \rangle F$ ("there is a terminating program run after which *F* holds")

Automotive Product Configuration (V)

- Translation of $\langle P \rangle F$ to propositional logic:
  $(x_1 \Rightarrow B_1) \wedge \ldots \wedge (x_n \Rightarrow B_n) \wedge (Z_1 \Rightarrow x_n) \wedge \ldots \wedge (Z_n \Rightarrow x_n) \wedge F$
  for complex formulae B_i, Z_i, F.
- Consistency criteria: PDL properties, encoding, e.g.:
  - Part p is never needed and thus superfluous.
  - There is an order for which the mutually exclusive parts p_1 and p_2 are simultaneously selected.
  - Equipment code x must not occur in any constructible order.
  - The result of the completion procedure differs, depending on the order of completion rule application.

Case Study II:

IBM System Automation - Expert System Verification

IBM System Automation (SA)

Automates operation of computer centers.

Keeps complex systems up and running:

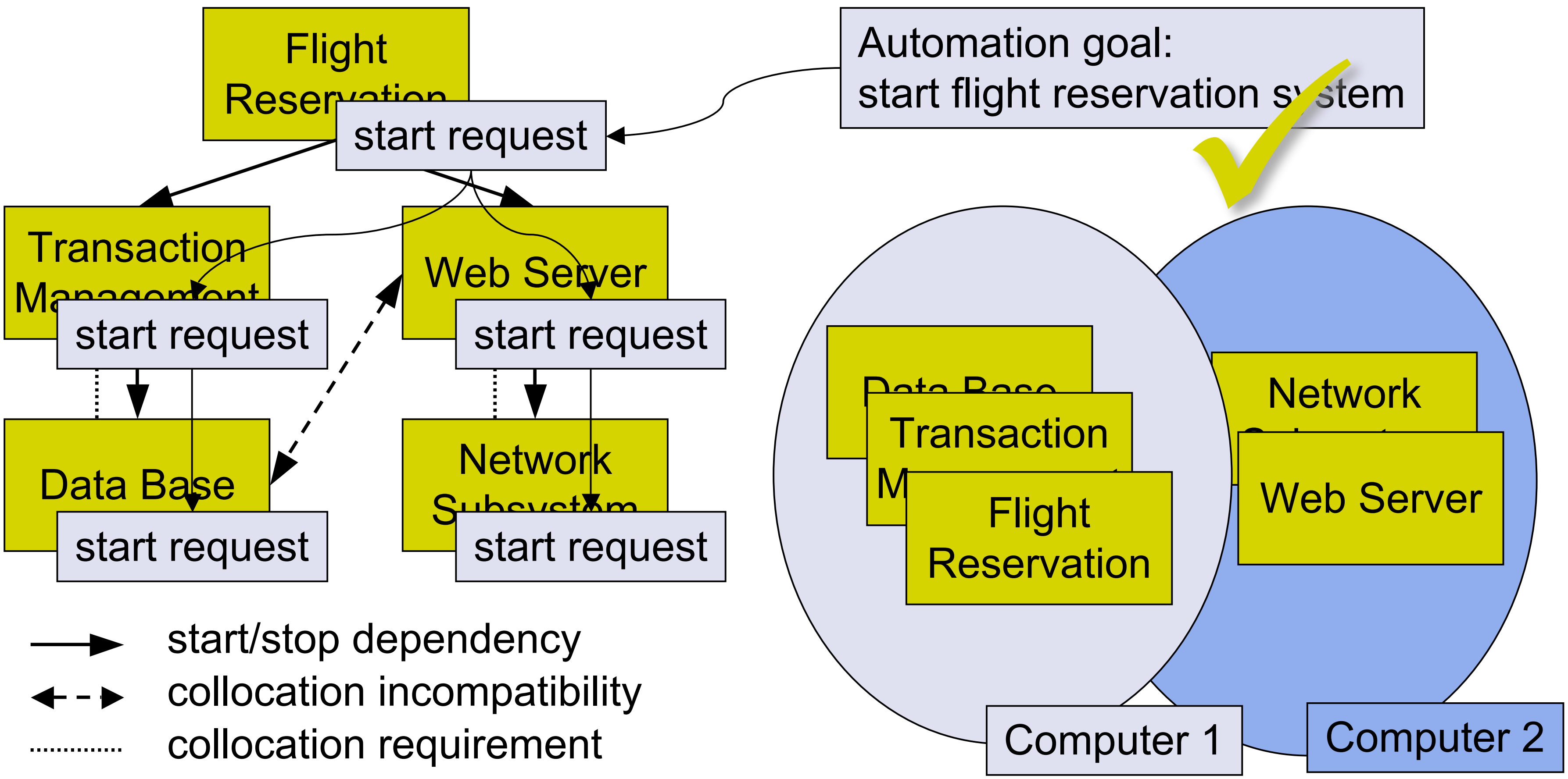
- Starting/stopping of applications
  (taking *dependencies* into account)
- Moving of applications between computers
  (e.g. on failure, for workload balancing)
- Supervision (active monitoring) of applications
  (current status? failure? system's workload?)
- Failure detection and error recovery (restart)

Verification of IBM's System Automation

- Rule-based expert system controls and monitors large sets of applications
  (starting, stopping, error recovery, load balancing, dependencies)
- Rules (finite-domain logic, WHEN-THEN) compute action sequence to reach given goal state
- Verified subsystem: 74 variables, 41 rules
- No cycles in computed action sequences?

  ⇒ **Propositional verification criteria (via intermediate language ΔPDL), SAT-checker, BDDs**

SA Example:
Flight Reservation System



System Automation: Rule Example

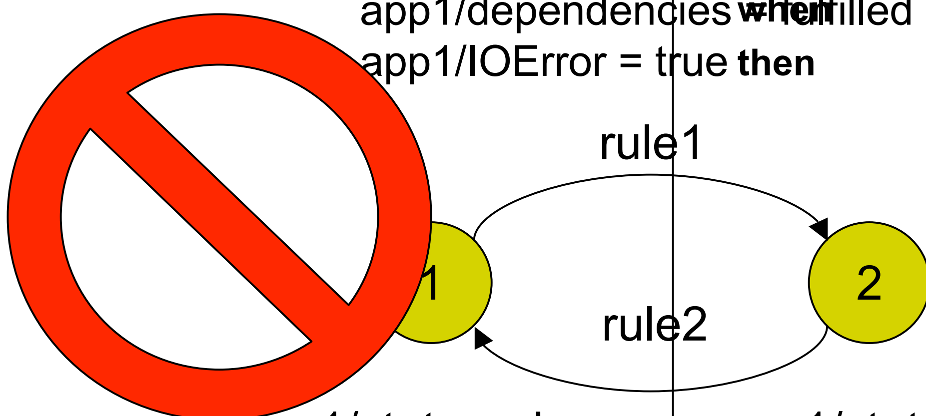
```
CORRELATION set/status/compound/satisfactory:
WHEN      status/compound NOT E {satisfactory}
    AND   status/startable E {yes}
    AND
    (     (          status/observed E {available, wasAvailable}
             AND     status/desired E {available}
             AND     status/automation E {idle, internal}
             AND     correlation/external/stop/failed E {false}
          )
          OR
          (          status/observed E {softDown, standBy}
             AND     status/desired E {unavailable}
             AND     status/automation E {idle, internal}
          )
    )
THEN      SetVariable status/compound = satisfactory
          RecordVariableHistory status/compound
```

SA's Expert System: Example

correlation rule1:
when app1/state = down
and app1/goal = up
and app1/dependencies = fulfilled
then app1/state = up

correlation rule2:
when app1/state = up
and app1/IOError = true
then app1/state = down

correlation rule3:
when app1/IOError = true
then app1/dependencies = pending



SA Verification Method

- Rule-execution program specified in PDL
- Termination property formulated in ΔPDL and converted to SAT
- Termination check by SAT checker
- In case of an error: Compute simplified result using BDDs (generalized error pattern)
- For details see

  Sinz, Lumpp, Schneider, Küchlin: **Detection of dynamic execution errors in IBM System Automation's rule-based expert system**. *Information and Software Technology*, 44(14):857-873, November 2002.

Implications on Logical Formalisms and Practical Experiences

Favorable Properties of Logical Formalism

- Support for finite domain variables
  - Groups of mutually exclusive variables very common in product configuration
  - Finite domain language already employed in IBM's rules

  ⇒ **Language of Boolean logic extended by selection operator $S_k(f_1,\ldots,f_n)$**
- Full formula structure
  - Conversion to CNF for large formula is time-consuming, increases formula size (or number of variables)

  ⇒ **No restriction to formulae in CNF**

Demands on Proof Procedure

- Support for extended propositional language

  ⇒ **Selection operator incorporated into Davis-Putnam-style algorithm for full propositional logic (no CNF)**

- Explanation
  - Indispensable for both proofs and failed proof attempts

  ⇒ **Proof explanation by generation of minimal unsatisfiable subformulae (MUS), counterexamples either by model generation (SAT) or BDDs**

  - Identification of generalized error patterns

  ⇒ **Distinction between relevant and irrelevant variables, existential abstraction over irrelevant variables (BDDs)**

Practical Experiences

- Surprisingly fast proofs in configuration domain
  - All proofs (formulae with >1000 propositional variables) by state-of-the-art SAT checker in <1 sec!

  ⇒ **Possible reason: always small conflicting rule sets, thus existence of short resolution proofs that carry over to DP**
- User's demands should be taken seriously
  - Prefer notions of problem domain to logical terminology
  - Graphical user interface, ease of use
  - Customized checks, as specialized as possible
  - Good integration into work-flow

Complexity Considerations

Complexity I: Experimental Results

- Automotive Configuration Domain:
  - Problem instances: base formula *B*, consistency condition *C*: Does $B \Rightarrow C$ hold? (Or equivalent: Is $B \wedge \neg C \in$ SAT?)
  - *B*: up to 1891 variables and 9947 clauses
  - >60'000 measured proofs with DP-style algorithm
- Results:
  - Surprisingly short search times: always <0.2s (decision tree: <600 branches)
  - Lemma generation *or* suitable variable selection heuristic in DP indispensable (here: *shortest positive clause*).
- Why did we obtain so good results?

Complexity II: Questions and Challenges

- Special problem structure?
  - Base formula *B* possesses many models
  - All Resolution proofs are very short (in case of UNSAT)
  - Saturation under Ordered Resolution feasible
- MUS (minimal unsatisfiable subset) computation:
  - Helpful for locating errors in large set of rules
  - Improved algorithms?

Summary

Two industrial case studies have shown similar results:

- Current SAT checking technology very powerful
- Adaptation of prover language and algorithms to industrial domains worthwhile
- Explanation of results (both positive and negative) important

For more information see

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