Product Configuration over the Internet

Presented at CIST 2001, Miami Beach, Florida

Jesper Møller [jm@it.edu]
Henrik Hulgaard [henrik@it.edu]
Henrik Reif Andersen [hra@it.edu]

The IT University of Copenhagen [www.it.edu]
ConfigIt Software [www.configit-software.com]

Product Configuration

- **Product model:** Variables $x_1, \ldots, x_n$, finite domains $D_1, \ldots, D_n$, rules $\phi_1, \ldots, \phi_m$.

Product Configuration

- **Product model:** Variables $x_1, \ldots, x_n$, finite domains $D_1, \ldots, D_n$, rules $\phi_1, \ldots, \phi_m$.
- **Solution:** An assignment of the variables such that each rule is satisfied.

Product Configuration

- **Product model:** Variables $x_1, \ldots, x_n$, finite domains $D_1, \ldots, D_n$, rules $\phi_1, \ldots, \phi_m$.
- **Solution:** An assignment of the variables such that each rule is satisfied.
- **Complexity:** Deciding consistency of a product model (is it satisfiable?) is NP-complete.

Product Configuration

- **Product model:** Variables $x_1, \ldots, x_n$, finite domains $D_1, \ldots, D_n$, rules $\phi_1, \ldots, \phi_m$.
- **Solution:** An assignment of the variables such that each rule is satisfied.
- **Complexity:** Deciding consistency of a product model (is it satisfiable?) is NP-complete.
- **Product configuration:**

Product Configuration

- **Product model:** Variables $x_1, \ldots, x_n$, finite domains $D_1, \ldots, D_n$, rules $\phi_1, \ldots, \phi_m$.
- **Solution:** An assignment of the variables such that each rule is satisfied.
- **Complexity:** Deciding consistency of a product model (is it satisfiable?) is NP-complete.
- **Product configuration:**
  - Is an assignment a solution?

Product Configuration

- **Product model:** Variables $x_1, \ldots, x_n$, finite domains $D_1, \ldots, D_n$, rules $\phi_1, \ldots, \phi_m$.
- **Solution:** An assignment of the variables such that each rule is satisfied.
- **Complexity:** Deciding consistency of a product model (is it satisfiable?) is NP-complete.
- **Product configuration:**
  - Is an assignment a solution?
  - Is a partial assignment consistent?

Product Configuration

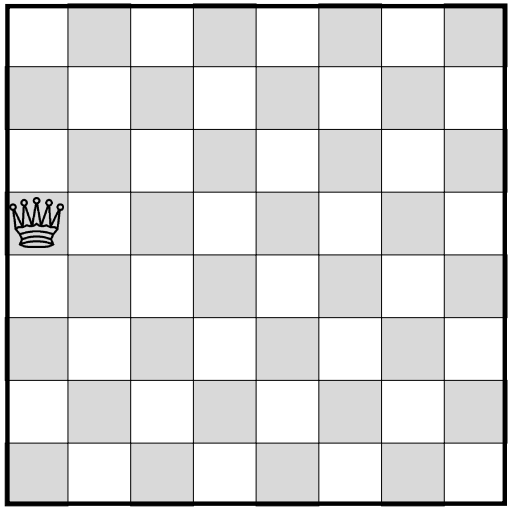
- **Product model:** Variables $x_1, \ldots, x_n$, finite domains $D_1, \ldots, D_n$, rules $\phi_1, \ldots, \phi_m$.
- **Solution:** An assignment of the variables such that each rule is satisfied.
- **Complexity:** Deciding consistency of a product model (is it satisfiable?) is NP-complete.
- **Product configuration:**
  - Is an assignment a solution?
  - Is a partial assignment consistent?
  - What values can be assigned to each variable, given a partial assignment?

Example: The 8-Queens Problem

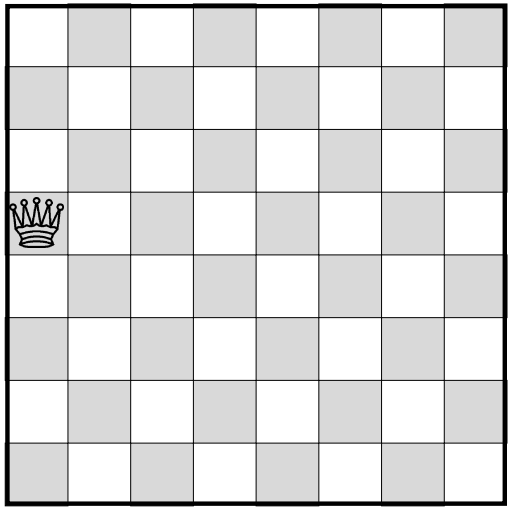
Place 8 queens on a chessboard so that no queen can capture another.

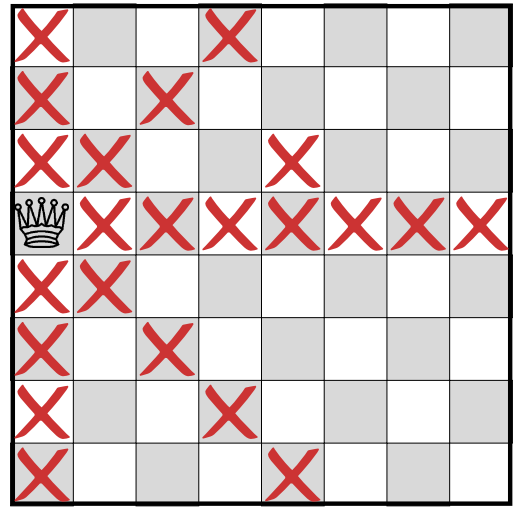
```
constant
   N: 8;
variable
   col: array N of [0..N-1];
rule
   forall i in [0..N-2]:
      (forall j in [i+1..N-1] :
         (col[i] <> col[j] and
          i+col[i] <> j+col[j] and
          i+col[j] <> j+col[i]);
```

Example: The 8-Queens Problem

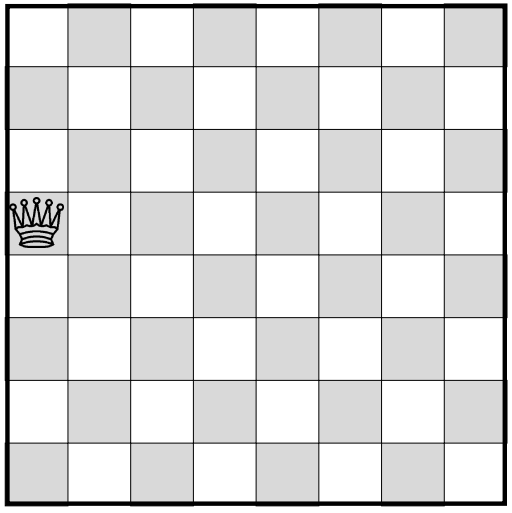


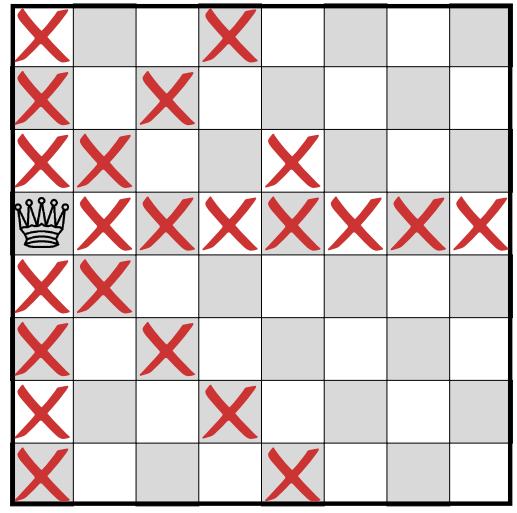
Example: The 8-Queens Problem

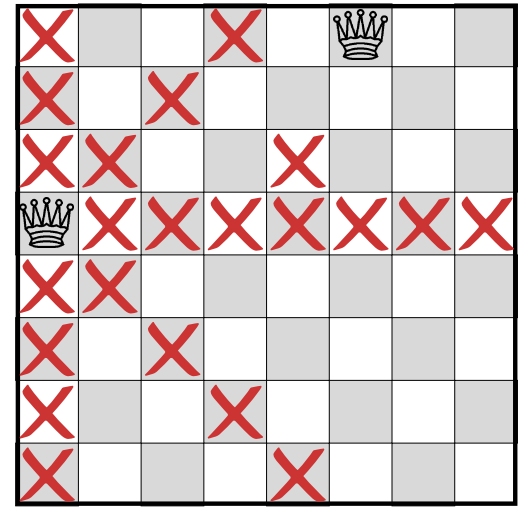




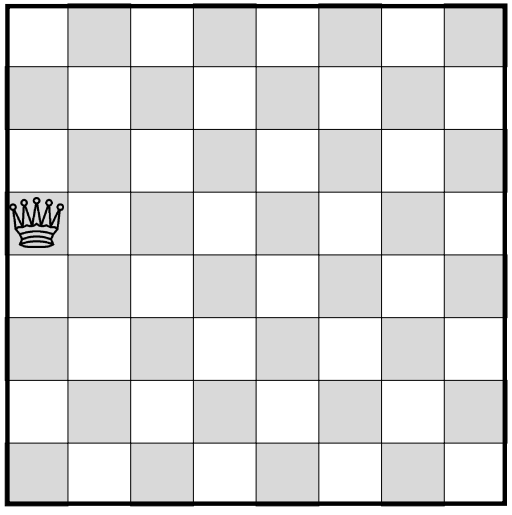
Example: The 8-Queens Problem

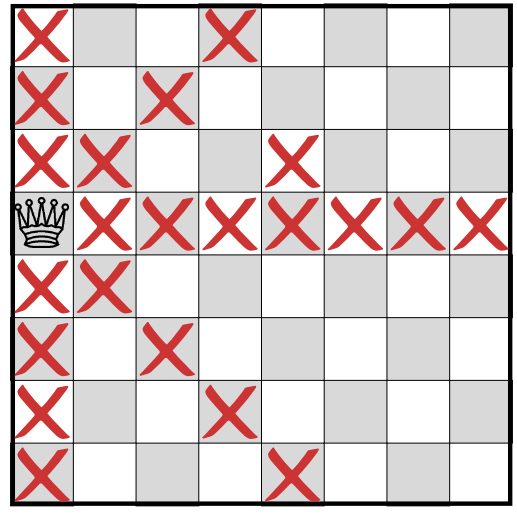


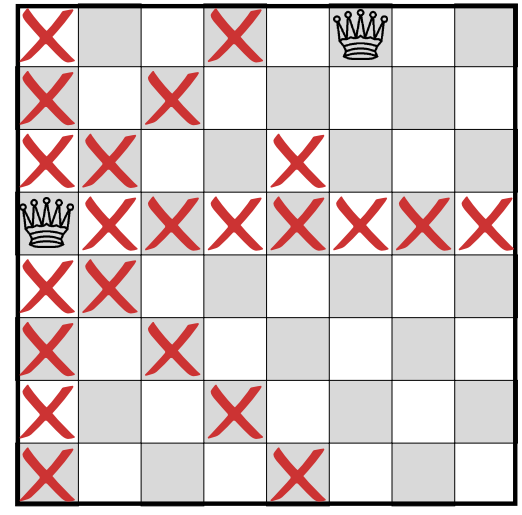


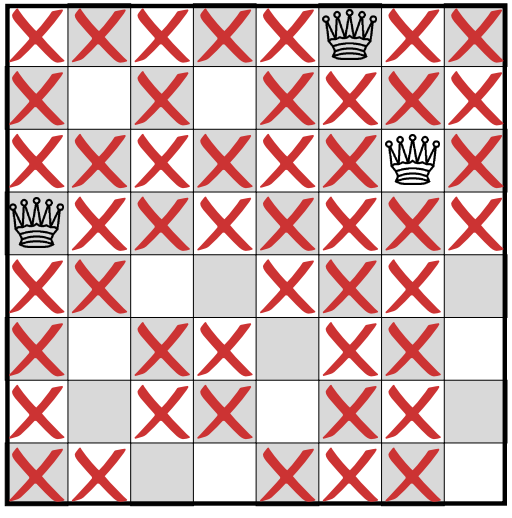


Example: The 8-Queens Problem

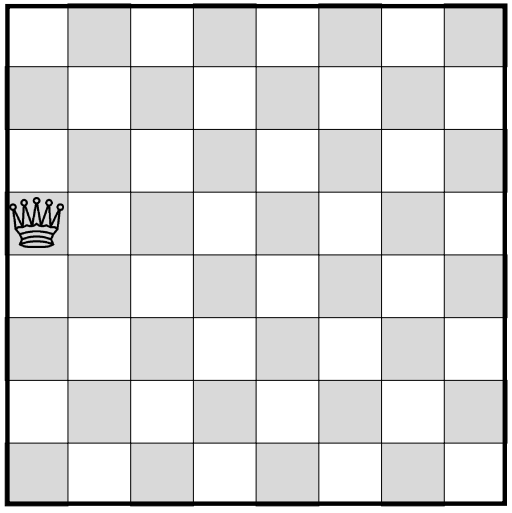


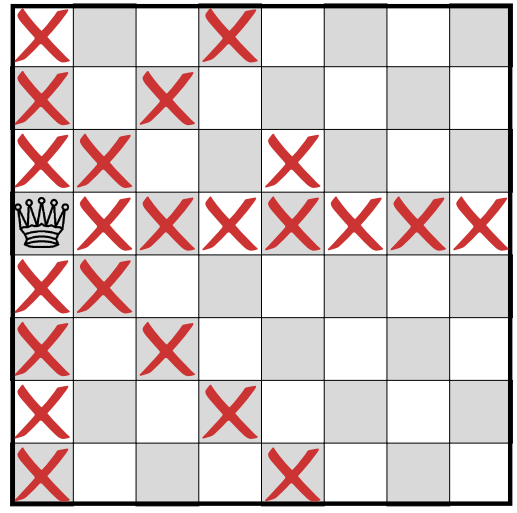


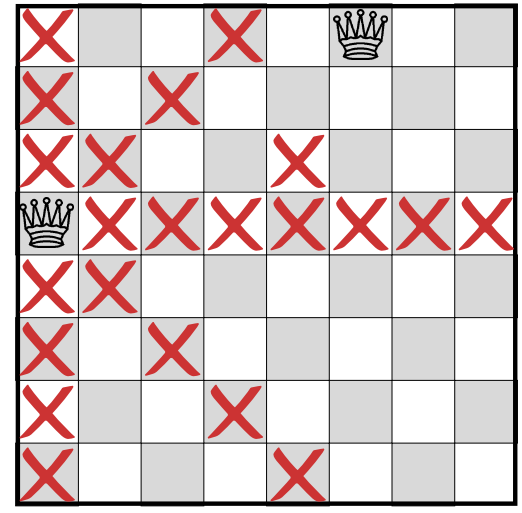


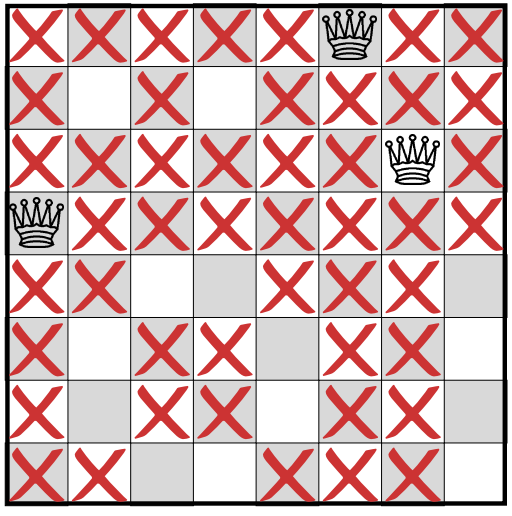


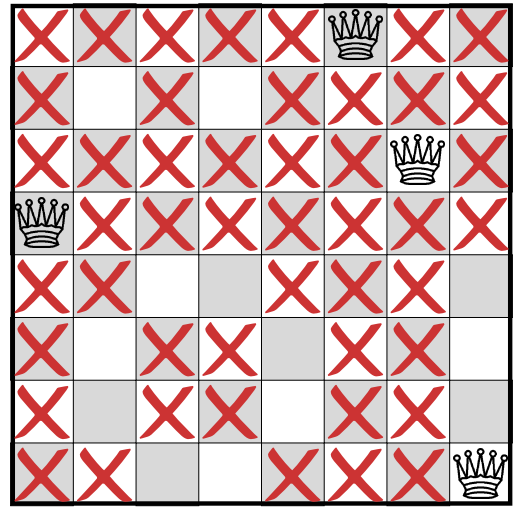
Example: The 8-Queens Problem



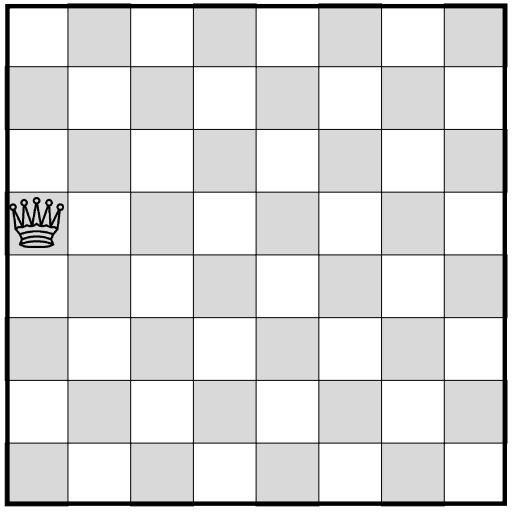


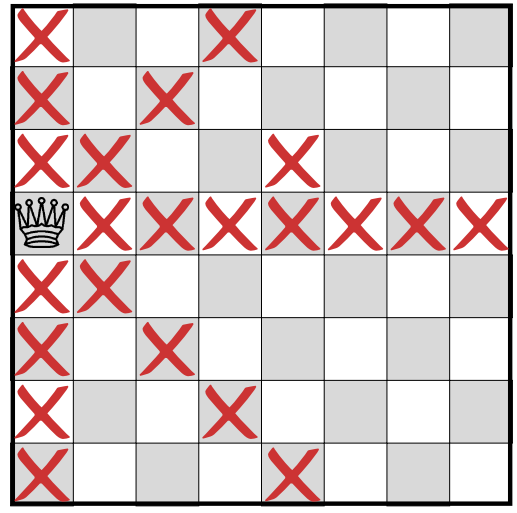


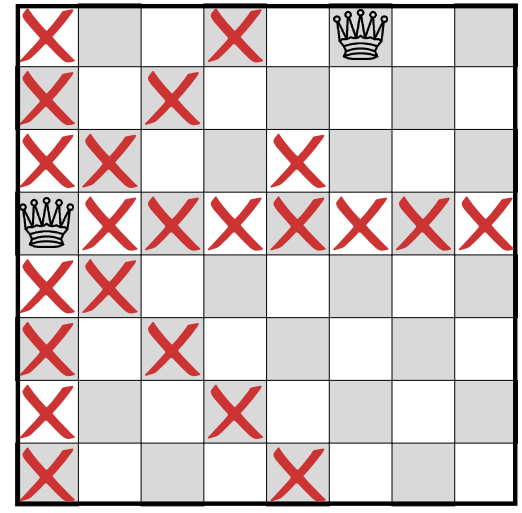


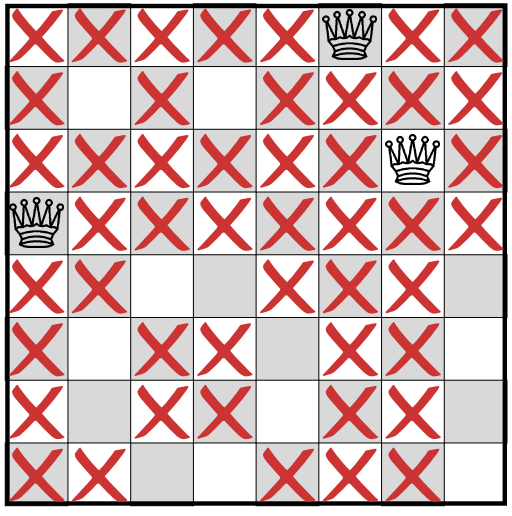


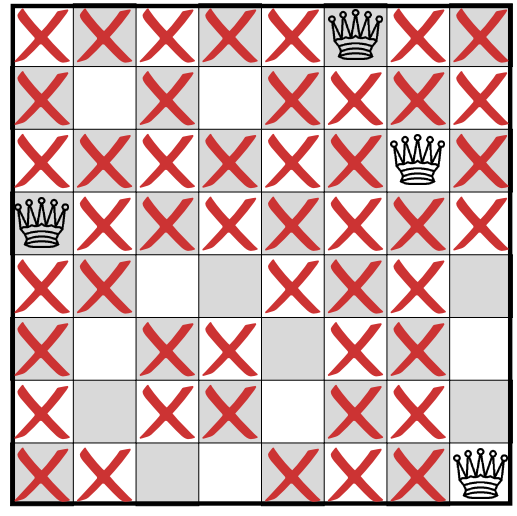
Example: The 8-Queens Problem

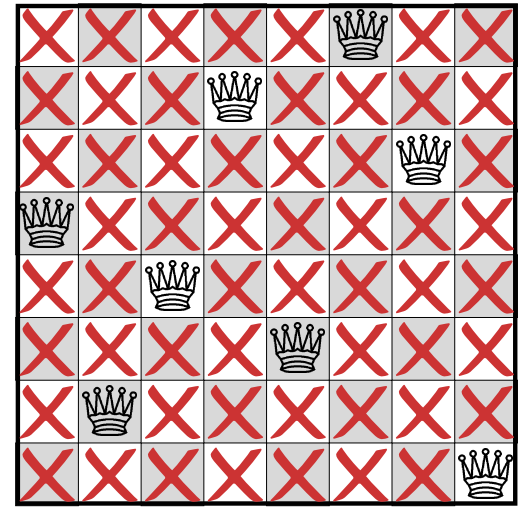












Constraint Propagation

- Whenever a user selects a value for a variable, derive the consequences by propagating this information through the rules.

Constraint Propagation

- Whenever a user selects a value for a variable, derive the consequences by propagating this information through the rules.
- Disadvantages:

Constraint Propagation

- Whenever a user selects a value for a variable, derive the consequences by propagating this information through the rules.
- Disadvantages:
  - response times may be too large, or

Constraint Propagation

- Whenever a user selects a value for a variable, derive the consequences by propagating this information through the rules.
- Disadvantages:
  - response times may be too large, or
  - only approximate solutions can be found

Constraint Propagation

- Whenever a user selects a value for a variable, derive the consequences by propagating this information through the rules.
- Disadvantages:
  - response times may be too large, or
  - only approximate solutions can be found
  - all work is done at runtime

Constraint Propagation

- Whenever a user selects a value for a variable, derive the consequences by propagating this information through the rules.
- Disadvantages:
  - response times may be too large, or
  - only approximate solutions can be found
  - all work is done at runtime
  - syntactic formulation of rules may influence response times

Constraint Propagation

- Whenever a user selects a value for a variable, derive the consequences by propagating this information through the rules.
- Disadvantages:
  - response times may be too large, or
  - only approximate solutions can be found
  - all work is done at runtime
  - syntactic formulation of rules may influence response times
  - difficult to reuse computations

Virtual Tabulation

- Compute all solution to the configuration problem and store them in a **virtual table**.

Virtual Tabulation

- Compute all solution to the configuration problem and store them in a **virtual table**.
- Virtual table: Compact data structure for representing all solutions implicitly.

Virtual Tabulation

- Compute all solution to the configuration problem and store them in a **virtual table**.
- Virtual table: Compact data structure for representing all solutions implicitly.
- Decision DAG where nodes are variables, and edges are values. A path = a solution.

Virtual Tabulation

- Compute all solution to the configuration problem and store them in a **virtual table**.
- Virtual table: Compact data structure for representing all solutions implicitly.
- Decision DAG where nodes are variables, and edges are values. A path = a solution.
- Size depends on the **structure** of the product model, not the number of solutions.

Virtual Tabulation

- Compute all solution to the configuration problem and store them in a **virtual table**.
- Virtual table: Compact data structure for representing all solutions implicitly.
- Decision DAG where nodes are variables, and edges are values. A path = a solution.
- Size depends on the **structure** of the product model, not the number of solutions.
- Efficient algorithms for building/querying a VT:
  Deciding consistency: $O(1)$.
  Possible values for x_i: $O(|D_i| + |\text{VT}|)$.

Virtual Tabulation

Advantages:

- Predictable, bounded response times

Virtual Tabulation

Advantages:

- Predictable, bounded response times
- All work is done at compile time

Virtual Tabulation

Advantages:

- Predictable, bounded response times
- All work is done at compile time
- The number of solutions is known

Virtual Tabulation

Advantages:

- Predictable, bounded response times
- All work is done at compile time
- The number of solutions is known
- Rules and other data need only be available at compile time

Virtual Tabulation

Advantages:

- Predictable, bounded response times
- All work is done at compile time
- The number of solutions is known
- Rules and other data need only be available at compile time
- Syntactic formulation of rules does not influence the size of the virtual table

Virtual Tabulation

Disadvantages:

- Only finite domains, no real-valued variables

Virtual Tabulation

Disadvantages:

- Only finite domains, no real-valued variables
- Relatively simple rules: plus, minus, comparison, and, or, not, if-then-else

Virtual Tabulation

Disadvantages:

- Only finite domains, no real-valued variables
- Relatively simple rules: plus, minus, comparison, and, or, not, if-then-else
- Virtual table must be recompiled whenever the product model changes

Virtual Tabulation

Disadvantages:

- Only finite domains, no real-valued variables
- Relatively simple rules: plus, minus, comparison, and, or, not, if-then-else
- Virtual table must be recompiled whenever the product model changes
- Most product models have compact virtual tables, but some don’t

Tool: ConfigIt Developer

- Product modeling language (PML): text/XML

Tool: ConfigIt Developer

- Product modeling language (PML): text/XML
- Product model compiler: PML $\longrightarrow$ VT

Tool: ConfigIt Developer

- Product modeling language (PML): text/XML
- Product model compiler: PML $\longrightarrow$ VT
- Product model viewer: stand-alone configurator

Tool: ConfigIt Developer

- Product modeling language (PML): text/XML
- Product model compiler: PML $\longrightarrow$ VT
- Product model viewer: stand-alone configurator
- Configuration server: COM-interface to a VT

Tool: ConfigIt Developer

- Product modeling language (PML): text/XML
- Product model compiler: PML $\longrightarrow$ VT
- Product model viewer: stand-alone configurator
- Configuration server: COM-interface to a VT
- Runs under Windows 2000/NT and Linux

Tool: ConfigIt Developer

- Product modeling language (PML): text/XML
- Product model compiler: PML $\longrightarrow$ VT
- Product model viewer: stand-alone configurator
- Configuration server: COM-interface to a VT
- Runs under Windows 2000/NT and Linux
- Available as add-on configuration module for IdealSeller [www.idealseller.com] developed by Catalog-International

Example: Customized Bike

Product model:

- Components: frame, tires, gear, pedals, etc.

Example: Customized Bike

Product model:

- Components: frame, tires, gear, pedals, etc.
- Properties: frame-size, frame-color, etc.

Example: Customized Bike

Product model:

- Components: frame, tires, gear, pedals, etc.
- Properties: frame-size, frame-color, etc.
- Data taken from a bike stores' SQL-database

Example: Customized Bike

Product model:

- Components: frame, tires, gear, pedals, etc.
- Properties: frame-size, frame-color, etc.
- Data taken from a bike stores' SQL-database
- User preferences: height, sex, type of bike

Example: Customized Bike

Product model:

- Components: frame, tires, gear, pedals, etc.
- Properties: frame-size, frame-color, etc.
- Data taken from a bike stores' SQL-database
- User preferences: height, sex, type of bike
- 42 variables, approx. 700 lines of rules

Example: Customized Bike

Product model:

- Components: frame, tires, gear, pedals, etc.
- Properties: frame-size, frame-color, etc.
- Data taken from a bike stores' SQL-database
- User preferences: height, sex, type of bike
- 42 variables, approx. 700 lines of rules
- Took 1 day to develop

Example: Customized Bike

Product model:

- Components: frame, tires, gear, pedals, etc.
- Properties: frame-size, frame-color, etc.
- Data taken from a bike stores' SQL-database
- User preferences: height, sex, type of bike
- 42 variables, approx. 700 lines of rules
- Took 1 day to develop
- 127 million solutions (different bikes)

Example: Customized Bike

Product model:

- Components: frame, tires, gear, pedals, etc.
- Properties: frame-size, frame-color, etc.
- Data taken from a bike stores' SQL-database
- User preferences: height, sex, type of bike
- 42 variables, approx. 700 lines of rules
- Took 1 day to develop
- 127 million solutions (different bikes)
- 1500 entries in VT

Example: Customized Bike

Web presentation:

- Use library of COM functions to query VT

Example: Customized Bike

Web presentation:

- Use library of COM functions to query VT
- To create a drop-down menu:

```
<% webControls.Dropdown "color", "(select)" %>
```

Example: Customized Bike

Web presentation:

- Use library of COM functions to query VT
- To create a drop-down menu:

```
<% webControls.Dropdown "color", "(select)" %>
```

- Approx. 200 lines of ASP

Example: Customized Bike

Web presentation:

- Use library of COM functions to query VT
- To create a drop-down menu:

```
<% webControls.Dropdown "color", "(select)" %>
```

- Approx. 200 lines of ASP
- Took 1–2 hours to develop

Example: Customized Bike

Web presentation:

- Use library of COM functions to query VT
- To create a drop-down menu:

```
<% webControls.Dropdown "color", "(select)" %>
```

- Approx. 200 lines of ASP
- Took 1–2 hours to develop
- Let's try it: [localhost/configit/Bike/bike.asp]

Contributions

- Virtual tabulation:

Contributions

- Virtual tabulation:
  - Predictable, bounded response times

Contributions

- Virtual tabulation:
  - Predictable, bounded response times
  - All work is done at compile time

Contributions

- Virtual tabulation:
  - Predictable, bounded response times
  - All work is done at compile time
  - Efficient graph-algorithms for building/querying a VT