Fault Localization in Constraint Programs

Nadjib LAZAAR*, Arnaud GOTLIEB*, Yahia LEBBAH**

* INRIA Rennes Bretagne Atlantique      ** Université d'Oran Es-Senia,

ICTAI’10, Arras, France
27 October
Motivations

- Numerous CP modeling languages and platforms have been developed (OPL, SICStus Prolog, ZINC, GECODE, CHOCO…) for solving combinatorial problems that arise in optimization, planning, scheduling…
Motivations

- Numerous CP modeling languages and platforms have been developed (OPL, SICStus Prolog, ZINC, GECODE, CHOCO…) for solving combinatorial problems that arise in optimization, planning, scheduling…
- CP programs begin to be used in business-critical systems (e.g., combinatorial auctions)
Motivations

- Numerous CP modeling languages and platforms have been developed (OPL, SICStus Prolog, ZINC, GECODE, CHOCO…)
  for solving combinatorial problems that arise in optimization, planning, scheduling…
- CP programs begin to be used in business-critical systems (e.g., combinatorial auctions)
- Refinement in CP
Motivations

- Numerous CP modeling languages and platforms have been developed (OPL, SICStus Prolog, ZINC, GECODE, CHOCO…) for solving combinatorial problems that arise in optimization, planning, scheduling…
- CP programs begin to be used in business-critical systems (e.g., combinatorial auctions)
- Refinement in CP
Motivations

- Numerous CP modeling languages and platforms have been developed (OPL, SICStus Prolog, ZINC, GECODE, CHOCO…) for solving combinatorial problems that arise in optimization, planning, scheduling…
- CP programs begin to be used in business-critical systems (e.g., combinatorial auctions)
- Refinement in CP
Motivations

- Numerous CP modeling languages and platforms have been developed (OPL, SICStus Prolog, ZINC, GECODE, CHOCO…)
  for solving combinatorial problems that arise in optimization, planning, scheduling…
- CP programs begin to be used in business-critical systems (e.g., combinatorial auctions)
- Refinement in CP
Motivations

- Numerous CP modeling languages and platforms have been developed (OPL, SICStus Prolog, ZINC, GECODE, CHOCO…) for solving combinatorial problems that arise in optimization, planning, scheduling…
- CP programs begin to be used in business-critical systems (e.g., combinatorial auctions)
- Refinement in CP
Golomb Rulers
Golomb Rulers
using CP;

int m=...;

dvar int x[1..m] in 0..m*m;

minimize x[m];

subject to {
  (1) forall (i in 1..m-1)
      x[i] < x[i+1];
  (2) forall (i in 1..m, j in 1..m,
           k in 1..m, l in 1..m:
               (i < j, k < l))
      x[j] - x[i] != x[l] - x[k];
}
Golomb Rulers in Optimization Programming Language

using CP;

int m=...;
dvar int x[1..m] in 0..m*m;
minimize x[m];
subject to {
  (1) forall (i in 1..m-1) x[i] < x[i+1];
  (2) forall (i in 1..m, j in 1..m, k in 1..m, l in 1..m: (i < j, k < l)) x[j] - x[i] != x[l] - x[k];
}

using CP;

int m=...;
tuple indexerTuple {int i; int j;}
{indexerTuple} indexes1 = {<i, j> | ordered i,j in 1..m};
{indexerTuple} indexes2 = {<i, j> | ordered i,j in 1..m div 2};
dvar int x[1..m] in 0..m*m;
dvar int d[indexes1];
minimize x[m];
subject to {
  (1) x[1]==0;
  (2) forall (i in 1..m-1) x[i] < x[i+1];
  (3) forall(ind in indexes1) d[ind] == x[ind.i]-x[ind.j];
  (4) x[m] >= (m * (m - 1)) / 2;
  (5) x[2] <= x[m]-x[m-1];
  (6) forall(ind1,ind2,ind3: (ind1.i==ind2.i) && (ind2.j==ind3.i) && (ind1.j==ind3.j) && (ind1.i<ind2.j<ind1.j))
      d[ind1] == d[ind2]+d[ind3];
  (7) forall(ind1,ind2,ind3,ind4 in indexes2 : (ind1.i==ind2.i) && (ind1.j==ind3.j) && (ind2.j==ind4.j)
      && (ind3.i==ind4.i) && (ind1.i<ind2.j && (3<ind1.j<ind2.j)) && (2<ind2.j<ind3.i<ind2.j <ind1.j)
      && (ind1.i<ind3.i<ind2.j <ind1.j))
      d[ind1] == (d[ind2]+d[ind3]-d[ind4]);
  (8) forall(ind in indexes2, k in 1..m div 2)
      (x[ind.i+1]==x[ind.i]+k) => (x[ind.j+1]!=x[ind.j]+k);
  }

M  P

Golomb Rulers in Optimization Programming Language

4 /18
1) Does P conform to M?
Golomb Rulers in Optimization Programming Language

using CP;
int m=...;
dvar int x[1..m] in 0..m*m;
minimize x[m];
subject to {
(1) forall (i in 1..m-1) x[i] < x[i+1];
(2) forall (i in 1..m, j in 1..m, k in 1..m, l in 1..m: (i < j, k < l)) x[j] - x[i] != x[l] - x[k];
}

1) Does P conform to M

m=8
X= [0 1 3 6 10 26 27 28]
using CP;

int m=...;

dvar int x[1..m] in 0..m*m;

minimize x[m];

subject to {
(1)  forall (i in 1..m-1)  x[i] < x[i+1];
(2)  forall (i in 1..m, j in 1..m, k in 1..m, l in 1..m: (i < j, k < l))
   x[j] - x[i] != x[l] - x[k];
}

1)- Does P conform to M

M

P

m=8

X= [0 1 3 6 10 26 27 28]

Fault detected in P!
using CP;
int m=...;
dvar int x[1..m] in 0..m*m;
minimize x[m];
subject to {
(1) forall (i in 1..m-1) x[i] < x[i+1];
(2) forall (i in 1..m, j in 1..m, k in 1..m, l in 1..m:
    (i < j, k < l)) x[j] - x[i] != x[l] - x[k];
}

1)- Does P conform to M

m=8

X= [0 1 3 6 10 26 27 28]

Fault detected in P !

2)- Where is the fault

using CP;
int m=...;
tuple indexerTuple {int i;
    int j;}
{indexerTuple} indexes1 = {<i, j> | ordered i,j in 1..m};
{indexerTuple} indexes2 = {<i, j> | ordered i,j in 1..m div 2};
dvar int x[1..m] in 0..m*m;
dvar int d[indexes1];
minimize x[m];
subject to {
(1) x[1]==0;
(2) forall (i in 1..m-1) x[i] < x[i+1];
(3) forall(ind in indexes1) d[ind] == x[ind.i]-x[ind.j];
(4) x[m] >= (m * (m - 1)) / 2;
(5) x[2] <= x[m]-x[m-1];
(6) forall(ind1,ind2,ind3:(ind1.i==ind2.i)&&(ind2.j==ind3.i)&&(ind3.j==ind1.i)&&
      (ind1.j==ind3.j)&&(ind1.i<ind2.j<ind1.j))
      d[ind1] == d[ind2]+d[ind3];
(7) forall(ind1,ind2,ind3,ind4 in indexes2 :
      (ind1.i==ind2.i)&&(ind1.j==ind3.j)&&(ind2.j==ind4.j)
      &&(ind3.i==ind4.i)&&(ind1.i<ind2.j<ind1.j)
      &&(2<ind2.j<m)&&(3<ind1.j<m+1)
      &&(2<ind3.i<m)&&(1<ind3.i<m-1)
      &&(ind1.i<ind3.i<ind2.j< ind1.j))
      d[ind1] == (d[ind2]+d[ind3]-d[ind4]);
(8) forall(ind in indexes2, k in 1..m div 2)
      (x[ind.i+1]==x[ind.i]+k) => (x[ind.j+1]!=x[ind.j]+k);
Golomb Rulers in Optimization Programming Language

using CP;
int m=...;
dvar int x[1..m] in 0..m*m;
minimize x[m];
subject to {
   (1) forall (i in 1..m-1)   x[i] < x[i+1];
   (2) forall (i in 1..m, j in 1..m, k in 1..m, l in 1..m:  
       (i < j, k < l))   
       x[j] - x[i] != x[l] - x[k];
}

1)- Does P conform to M

m=8

\[ X = [0 \ 1 \ 3 \ 6 \ 10 \ 26 \ 27 \ 28] \]

Fault detected in P !

using CP;
int m=...;
tuple indexerTuple {int i;         
   int j;}
{indexerTuple} indexes1 = {<i, j> | ordered i,j in 1..m};
{indexerTuple} indexes2 = {<i, j> | ordered i,j in 1..m div 2};
dvar int x[1..m] in 0..m*m;
dvar int d[indexes1];
minimize x[m];
subject to {
   (1) x[1]==0;
   (2) forall (i in 1..m-1)   x[i] < x[i+1];
   (3) forall(ind in indexes1)   d[ind] == x[ind.i]-x[ind.j];
   (4) x[m] >= (m * (m - 1)) / 2;
   (5) x[2] <= x[m]-x[m-1];
   (6) forall(ind1,ind2,ind3:   (ind1.i==ind2.i)&&(ind2.j==ind3.i)&&(ind1.j==ind3.j)&&(ind1.i<ind2.j<ind1.j))
       d[ind1] == d[ind2]+d[ind3];
   (7) forall(ind1,ind2,ind3,ind4 in indexes2 :   (ind1.i==ind2.i)&&(ind1.j==ind3.j)&&(ind2.j==ind4.j) 
       &&(ind3.i==ind4.i)&&(ind1.i<m-1)&&(3<ind1.j<m+1) 
       &&(2<ind2.j<m)&&(1<ind3.i<m-1) 
       &&(ind1.i<ind3.i<ind2.j< ind1.j))
       d[ind1] == (d[ind2]+d[ind3]-d[ind4]);
   (8) forall(ind in indexes2, k in 1..m div 2)
       (x[ind.i+1]==x[ind.i]+k) => (x[ind.j+1]!=x[ind.j]+k);
}

2)-Where is the fault

Faulty constraint!!
Contributions

• A first framework for fault localization in constraint programs
  • Context
  • Fault localization process
Contributions

• A first framework for fault localization in constraint programs
  • Context
  • Fault localization process

• A method and a tool, called CPTEST:
  • to detect non-conformities (e.g., \(X = [0, 1, 3, 6, 10, 26, 27, 28]\))
  • to localize faulty constraints
  • short demonstration of CPTEST
Contributions

• A first framework for fault localization in constraint programs
  • Context
  • Fault localization process

• A method and a tool, called CPTEST:
  • to detect non-conformities (e.g., \(X = [0, 1, 3, 6, 10, 26, 27, 28]\))
  • to localize faulty constraints
  • short demonstration of CPTEST

• An experimental validation on classical benchmarks
  (Golomb rulers, car sequencing, n-queens, social golfer)
M implements the problem specification (testing oracle)
The underlying constraint solver is correct
Context

- M implements the problem specification (testing oracle)
- The underlying constraint solver is correct
Context

Model-Oracle M

Constraint solving problem

\[ M \equiv C_1 \land C_2 \ldots \land C_n \]

sol(M) : set of solution of M

CPUT P

Optimization problem

\[ P \equiv C'_1 \land C'_2 \ldots \land C'_m \]

sol(P) : set of solution of P

- \( M \) implements the problem specification (testing oracle)
- The underlying constraint solver is correct
Context*

- Conformity relation

one solution:

\[ P \text{ con}\!f\!M \iff \text{sol}(P) \neq \emptyset \land \text{sol}(P) \subseteq \text{sol}(M) \]
Context

- Conformity relation

one solution:

\[ P \text{ con}f M \iff \text{sol}(P) \neq \emptyset \land \text{sol}(P) \subseteq \text{sol}(M) \]

all solutions:

\[ P \text{ con}f M \iff \text{sol}(P) = \text{sol}(M) \]

Context*

- Conformity relation

one solution:

\[ P \text{ confM} \iff \text{sol}(P) \neq \emptyset \land \text{sol}(P) \subseteq \text{sol}(M) \]

all solutions:

\[ P \text{ confM} \iff \text{sol}(P) = \text{sol}(M) \]

best solution:

\[ P \text{ confM} \iff \text{bounds}_{lw,up,f'}(P) \neq \emptyset \land \text{bounds}_{lw,up,f'}(P) \subseteq \text{bounds}_{lw,up,f}(M) \]

Fault localization (suspicious constraints)

$C_i$ is suspicious in $P$ w.r.t. $M \equiv M \land \{C_1, C_2, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n\}$ is satisfiable.

$P \setminus \{C_i\}$
Fault localization (suspicious constraints)

\[ C_i \text{ is suspicious in } P \text{ w.r.t. } M \equiv M \land \{C_1, C_2, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n\} \text{ is satisfiable.} \]
\[ P \setminus \{C_i\} \]

Objective: \( \text{SuspiciousSet}(P) \equiv \{C \in P : C \text{ is suspicious in } P \text{ w.r.t. } M\} \)
Fault localization (suspicous constraints)

$C_i$ is suspicious in $P$ w.r.t. $M \equiv M \land \{C_1, C_2, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n\}$ is satisfiable.

$P \setminus \{C_i\}$

Objective: $\text{SuspiciousSet}(P) \equiv \{C \in P : C$ is suspicious in $P$ w.r.t. $M\}$

$\implies \text{SuspiciousSet}(P) = \{C_i\} \implies$ the singleton $C_i$ is the faulty constraint
Fault localization (suspicious constraints)

$C_i$ is suspicious in $P$ w.r.t. $M \equiv M \setminus \{C_1, C_2, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n\}$ is satisfiable. $P \setminus \{C_i\}$

Objective: $SuspicousSet(P) \equiv \{C \in P : C \text{ is suspicious in } P \text{ w.r.t. } M\}$

- $SuspicousSet(P) = \{C_i\} \Rightarrow$ the singleton $C_i$ is the faulty constraint
- $SuspicousSet(P) = \{C_j, \ldots, C_k\} \Rightarrow C_j, \ldots, C_k$ are suspicious constraints
Fault localization (suspicious constraints)

\( C_i \) is suspicious in \( P \) w.r.t. \( M \equiv M \land \{C_1, C_2, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n\} \) is satisfiable. \( P \setminus \{C_i\} \)

Objective: \( \text{SuspiciousSet}(P) \equiv \{C \in P : C \text{ is suspicious in } P \text{ w.r.t. } M\} \)

- \( \text{SuspiciousSet}(P) = \{C_i\} \Rightarrow \text{the singleton } C_i \text{ is the faulty constraint} \)
- \( \text{SuspiciousSet}(P) = \{C_j, \ldots, C_k\} \Rightarrow C_j \ldots C_k \text{ are suspicious constraints} \)
- \( \text{SuspiciousSet}(P) = P \Rightarrow \text{calculate addSet form } M \)
Fault localization (suspicious constraints)

$C_i$ is suspicious in $P$ w.r.t. $M \equiv M \land \{C_1, C_2, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n\}$ is satisfiable.

Objective: $\text{SuspiciousSet}(P) \equiv \{C \in P : C \text{ is suspicious in } P \text{ w.r.t. } M\}$

- $\text{SuspiciousSet}(P) = \{C_i\} \Rightarrow$ the singleton $C_i$ is the faulty constraint
- $\text{SuspiciousSet}(P) = \{C_j, \ldots, C_k\} \Rightarrow$ $C_j \ldots C_k$ are suspicious constraints
- $\text{SuspiciousSet}(P) = P \Rightarrow$ calculate $\text{addSet}$ form $M$

Hypothesis: $P$ contains one faulty constraint
Fault localization (intuitions)

$\text{SuspiciousSet}(P) = \emptyset$

$\text{addSet} = \emptyset$

- **M**: set of solution of the Model-Oracle
- **C1**: set of solution of constraint C1
- **C2**: set of solution of constraint C2
- **C3**: set of solution of constraint C3
Fault localization (intuitions)

$$\text{sol}(M \land P/\{C_1\}) = \emptyset \quad C_1 \text{ is not a suspicious constraint}$$

$\textbf{M}$: set of solution of the Model-Oracle
$\textbf{C}_1$: set of solution of constraint $C_1$
$\textbf{C}_2$: set of solution of constraint $C_2$
$\textbf{C}_3$: set of solution of constraint $C_3$
Fault localization (intuitions)

\[ sol(M \land P/\{C_1\}) = \emptyset \]

\[ sol(M \land P/\{C_2\}) = \emptyset \quad C_2 \text{ is not a suspicious constraint} \]

\[ \text{SuspiciousSet}(P) = \emptyset \]

\[ \text{addSet} = \emptyset \]

\textbf{M}: set of solution of the Model-Oracle

\textbf{C1}: set of solution of constraint C1

\textbf{C2}: set of solution of constraint C2

\textbf{C3}: set of solution of constraint C3
Fault localization (intuitions)

\[
\text{sol}(M \land P/\{C_1\}) = \emptyset
\]

\[
\text{sol}(M \land P/\{C_2\}) = \emptyset
\]

\[
\text{sol}(M \land P/\{C_3\}) \neq \emptyset \quad C_3 \text{ is a suspicious constraint}
\]

\[
\text{SuspiciousSet}(P) = \{C_3\}
\]

\[
\text{addSet} = \emptyset
\]

\[M: \text{set of solution of the Model-Oracle}\]
\[C_1: \text{set of solution of constraint C1}\]
\[C_2: \text{set of solution of constraint C2}\]
\[C_3: \text{set of solution of constraint C3}\]

\[C_3 \text{ is the faulty constraint!!}\]
Fault localization (intuitions)

\[ \text{SuspiciousSet}(P) = \emptyset \]
\[ \text{addSet} = \emptyset \]

**M**: set of solution of the Model-Oracle

**C1**: set of solution of constraint C1

**C2**: set of solution of constraint C2

**C3**: set of solution of constraint C3
Fault localization (intuitions)

\[ \text{sol}(M \land P/\{C_1\}) \neq \emptyset \quad \text{C}_1 \text{ is a suspicious constraint} \]
\[ \text{sol}(M \land P/\{C_2\}) \neq \emptyset \quad \text{C}_2 \text{ is a suspicious constraint} \]
\[ \text{sol}(M \land P/\{C_3\}) \neq \emptyset \quad \text{C}_3 \text{ is a suspicious constraint} \]

\[ \text{SuspiciousSet}(P) = \{C_1, C_2, C_3\} \]
\[ \text{addSet} = \emptyset \]

**M**: set of solution of the Model-Oracle

**C1**: set of solution of constraint C1

**C2**: set of solution of constraint C2

**C3**: set of solution of constraint C3
Fault localization (intuitions)

\[ \text{sol}(M \land P/\{C_1\}) \neq \emptyset \quad \text{C}_1 \text{ is a suspicious constraint} \]
\[ \text{sol}(M \land P/\{C_2\}) \neq \emptyset \quad \text{C}_2 \text{ is a suspicious constraint} \]
\[ \text{sol}(M \land P/\{C_3\}) \neq \emptyset \quad \text{C}_3 \text{ is a suspicious constraint} \]

\[ \text{SuspiciousSet}(P) = \{C_1, C_2, C_3\} \]
\[ \text{addSet} = \{C'\} \]

**M:** set of solution of the Model-Oracle  
**C1:** set of solution of constraint C1  
**C2:** set of solution of constraint C2  
**C3:** set of solution of constraint C3
Fault localization (algorithm)

Algorithm: locate(M, P, nc)

SuspiciousSet ← ∅
addSet ← ∅

foreach C_i ∈ P do
    if sol(M ∩ P\{C_i}) ≠ ∅ then
        SuspiciousSet ← SuspiciousSet ∪ \{C_i\}
    fi
esac

if (SuspiciousSet = P) ∧ (nc ≠ ∅) then
    addSet ← checking(M, nc)
fi

return (rev(SuspiciousSet), addSet)
Fault localization (algorithm)

Algorithm: locate(M, P, nc)

\[
\begin{align*}
\text{SuspiciousSet} & \leftarrow \emptyset \\
\text{addSet} & \leftarrow \emptyset \\
\text{foreach } C_i \in P \text{ do} & \\
& \quad \text{if } \text{sol}(M \land P\setminus\{C_i\}) \neq \emptyset \text{ then} \\
& \qquad \text{SuspiciousSet} \leftarrow \text{SuspiciousSet} \cup \{C_i\} \\
\text{if } (\text{SuspiciousSet} = P) \land (\text{nc} \neq \emptyset) \text{ then} & \\
& \quad \text{addSet} \leftarrow \textbf{checking}(M, \text{nc}) \\
\text{return } (\text{rev(SuspiciousSet)}, \text{addSet})
\end{align*}
\]

**checking(B, nc):**

\[
\begin{align*}
\text{if } B = \emptyset \text{ then} & \\
& \quad \text{return } \emptyset \\
\text{else} & \\
& \quad \text{foreach } C_i \in B \text{ do} \\
& \qquad \text{if } \text{nc} \models C_i \text{ then} \\
& \quad \qquad \text{return } \textbf{checking}(M\setminus\{C_i\}, \text{nc}) \\
& \qquad \text{else} \\
& \quad \qquad \text{return } \{C_i\} \cup \textbf{checking}(M\setminus\{C_i\}, \text{nc})
\end{align*}
\]
CPTEST tool
Manual fault injection

using CP;

int m = ...;
tuple indexerTuple {int i;
  int j;}
{indexerTuple} indexes1 = {{i, j} | ordered i,j in 1..m};
{indexerTuple} indexes2 = {{i, j} | ordered i,j in 1..m div 2};
dvar int x[1..m] in 0..m*m;
dvar int d[indexes1];

minimize x[m];

subject to {
(1) x[1] == 0;
(2) forall (i in 1..m-1) x[i] < x[i+1];
(3) forall (ind in indexes1) d[ind] == x[ind.i] - x[ind.j];
(4) x[m] >= (m * (m - 1)) / 2;
(5) allDifferent(all(ind in indexes1) d[ind]);
(6) x[2] <= x[m] - x[m-1];
(7) forall (ind1,ind2,ind3: (ind1.i==ind2.i) && (ind2.j==ind3.i) &&
  (ind1.j==ind3.j) && (ind1.i<ind2.j<ind1.j))
    d[ind1] == d[ind2] + d[ind3];
(8) forall (ind1,ind2,ind3,ind4 in indexes2 : 
    (ind1.i==ind2.i) && (ind1.j==ind3.j) && (ind2.j==ind4.j) &&
    (ind3.i==ind4.i) && (ind1.i<ind3.i) && (3<ind1.j<m+1) &&
    (2<ind2.j<m) && (1<ind3.i<m-1) &&
    (ind1.i<ind3.i<ind2.j<ind1.j))
    d[ind1] == (d[ind2] + d[ind3] - d[ind4]);
(9) forall (ind in indexes2, k in 1..m div 2)
    (x[ind.i+1] == x[ind.i]+k) => (x[ind.j+1] != x[ind.j]+k);
}

Validation (Golomb Rulers)
Validation (Golomb Rulers)

- Manual fault injection

```cpp
using CP;

int m=...;
tuple indexerTuple {int i;
    int j;}
{indexerTuple} indexes1 = {<i, j> | ordered i,j in 1..m};
{indexerTuple} indexes2 = {<i, j> | ordered i,j in 1..m div 2};
dvar int x[1..m] in 0..m*m;
dvar int d[indexes1];

minimize x[m];

subject to {
(1) x[1]==0;
(2) forall (i in 1..m-1)   x[i] < x[i+1];
(3) forall(ind in indexes1)   d[ind] == x[ind.i] - x[ind.j];
(4) x[m] >= (m * (m - 1)) / 2;
(5) allDifferent(all(ind in indexes1 ) d[ind]);
(6) x[2] <= x[m] - x[m-1];
(7) forall(ind1,ind2,ind3: (ind1.i==ind2.i)&&(ind2.j==ind3.i)&&(ind1.j==ind3.j)&&(ind1.i<ind2.j<ind1.j))
    d[ind1] == d[ind2] + d[ind3];
(8) forall(ind1,ind2,ind3,ind4 in indexes2 : 
    (ind1.i==ind2.i)&&(ind1.j==ind3.j)&&(ind2.j==ind4.j) &&(ind3.i==ind4.i)&&(ind1.i<m-1)&&(3<ind1.j<m+1) &&(2<ind2.j<m)&&(1<ind3.i<m-1) &&
    (ind1.i<ind3.i<ind2.j<ind1.j))
    d[ind1] == (d[ind2] + d[ind3] - d[ind4]);
(9) forall(ind in indexes2, k in 1..m div 2)
    (x[ind.i+1] == x[ind.i]+k) => (x[ind.j+1] != x[ind.j]+k);
    }
```

forall(i in m..3*m)
    count(all(j in indexes1) d[j], i) == 1;
Validation (Golomb Rulers)

<table>
<thead>
<tr>
<th>Test</th>
<th>Golomb rulers</th>
<th># consts</th>
<th>fault injected</th>
<th>non-conformity detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>2</td>
<td>p1_ct2</td>
<td></td>
<td>[0 1 3 6]</td>
</tr>
<tr>
<td>CPUT2</td>
<td>4</td>
<td>p2_ct3</td>
<td></td>
<td>[0 1 6 11]</td>
</tr>
<tr>
<td>CPUT3</td>
<td>5</td>
<td>p3_ct3</td>
<td></td>
<td>[0 5 6 11]</td>
</tr>
<tr>
<td>CPUT4</td>
<td>9</td>
<td>p4_ct5</td>
<td></td>
<td>(\text{sol}(P4) = \emptyset)</td>
</tr>
<tr>
<td>CPUT5</td>
<td>9</td>
<td>p5_ct1</td>
<td></td>
<td>(\text{sol}(P5) = \emptyset)</td>
</tr>
<tr>
<td>CPUT6</td>
<td>9</td>
<td>p6_ct2</td>
<td></td>
<td>[0 1 6 11]</td>
</tr>
<tr>
<td>CPUT7</td>
<td>9</td>
<td>p7_ct9</td>
<td></td>
<td>(\text{sol}(P7) = \emptyset)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Localization</th>
<th>Golomb rulers</th>
<th>fault localized</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>(rev(P1), (m_ct2))</td>
<td>0.32s</td>
<td></td>
</tr>
<tr>
<td>CPUT2</td>
<td>(rev(P2), (m_ct2))</td>
<td>0.82s</td>
<td></td>
</tr>
<tr>
<td>CPUT3</td>
<td>(rev(P3), (m_ct2))</td>
<td>1.15s</td>
<td></td>
</tr>
<tr>
<td>CPUT4</td>
<td>(rev(p4_ct5),\emptyset)</td>
<td>1.87s</td>
<td></td>
</tr>
<tr>
<td>CPUT5</td>
<td>(rev(p5_ct1),\emptyset)</td>
<td>1.62s</td>
<td></td>
</tr>
<tr>
<td>CPUT6</td>
<td>(rev(P6), (m_ct2))</td>
<td>2.14s</td>
<td></td>
</tr>
<tr>
<td>CPUT7</td>
<td>(rev(p7_ct6, p7_ct9),\emptyset)</td>
<td>1.92s</td>
<td></td>
</tr>
</tbody>
</table>
## Validation (Golomb Rulers)

<table>
<thead>
<tr>
<th>Test</th>
<th>Golomb rulers</th>
<th># consts</th>
<th>fault injected</th>
<th>non-conformity detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>2</td>
<td>p1_ct2</td>
<td>[0 1 3 6]</td>
<td></td>
</tr>
<tr>
<td>CPUT2</td>
<td>4</td>
<td>p2_ct3</td>
<td>[0 1 6 11]</td>
<td></td>
</tr>
<tr>
<td>CPUT3</td>
<td>5</td>
<td>p3_ct3</td>
<td>[0 5 6 11]</td>
<td></td>
</tr>
<tr>
<td>CPUT4</td>
<td>9</td>
<td>p4_ct5</td>
<td>sol(P4) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT5</td>
<td>9</td>
<td>p5_ct1</td>
<td>sol(P5) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT6</td>
<td>9</td>
<td>p6_ct2</td>
<td>sol(P7) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT7</td>
<td>9</td>
<td>p7_ct9</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Localization</th>
<th>fault localized</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>(rev(P1), (m_ct2))</td>
<td>0.32s</td>
</tr>
<tr>
<td>CPUT2</td>
<td>(rev(P2), (m_ct2))</td>
<td>0.82s</td>
</tr>
<tr>
<td>CPUT3</td>
<td>(rev(P3), (m_ct2))</td>
<td>1.15s</td>
</tr>
<tr>
<td>CPUT4</td>
<td>(rev(p4_ct5), ∅)</td>
<td>1.87s</td>
</tr>
<tr>
<td>CPUT5</td>
<td>(rev(p5_ct1), ∅)</td>
<td>1.62s</td>
</tr>
<tr>
<td>CPUT6</td>
<td>(rev(P6), (m_ct2))</td>
<td>2.14s</td>
</tr>
<tr>
<td>CPUT7</td>
<td>(rev(p7_ct6, p7_ct9), ∅)</td>
<td>1.92s</td>
</tr>
</tbody>
</table>

Intel Core 2 Duo CPU, 2.39 GHz, 2.00 Go RAM
# Validation (Golomb Rulers)

<table>
<thead>
<tr>
<th>Test</th>
<th>Golomb rulers</th>
<th># consts</th>
<th>fault injected</th>
<th>non-conformity detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>2</td>
<td>p1_ct2</td>
<td>[0 1 3 6]</td>
<td></td>
</tr>
<tr>
<td>CPUT2</td>
<td>4</td>
<td>p2_ct3</td>
<td>[0 1 6 11]</td>
<td></td>
</tr>
<tr>
<td>CPUT3</td>
<td>5</td>
<td>p3_ct3</td>
<td>[0 5 6 11]</td>
<td></td>
</tr>
<tr>
<td><strong>CPUT4</strong></td>
<td><strong>9</strong></td>
<td><strong>p4_ct5</strong></td>
<td><strong>sol(P4) = ∅</strong></td>
<td></td>
</tr>
<tr>
<td>CPUT5</td>
<td>9</td>
<td>p5_ct1</td>
<td>sol(P5) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT6</td>
<td>9</td>
<td>p6_ct2</td>
<td>[0 1 6 11]</td>
<td></td>
</tr>
<tr>
<td>CPUT7</td>
<td>9</td>
<td>p7_ct9</td>
<td>sol(P7) = ∅</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Localization</th>
<th>Golomb rulers</th>
<th>fault localized</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>(rev(P1), (m_ct2))</td>
<td>0.32s</td>
<td></td>
</tr>
<tr>
<td>CPUT2</td>
<td>(rev(P2), (m_ct2))</td>
<td>0.82s</td>
<td></td>
</tr>
<tr>
<td>CPUT3</td>
<td>(rev(P3), (m_ct2))</td>
<td>1.15s</td>
<td></td>
</tr>
<tr>
<td><strong>CPUT4</strong></td>
<td>(rev(p4_ct5), ∅)</td>
<td><strong>1.87s</strong></td>
<td></td>
</tr>
<tr>
<td>CPUT5</td>
<td>(rev(p5_ct1), ∅)</td>
<td>1.62s</td>
<td></td>
</tr>
<tr>
<td>CPUT6</td>
<td>(rev(P6), (m_ct2))</td>
<td>2.14s</td>
<td></td>
</tr>
<tr>
<td>CPUT7</td>
<td>(rev(p7_ct6, p7_ct9), ∅)</td>
<td>1.92s</td>
<td></td>
</tr>
</tbody>
</table>

Intel Core 2 Duo CPU, 2.39 GHz, 2.00 Go RAM
Validation (Golomb Rulers)

<table>
<thead>
<tr>
<th>Test</th>
<th>Golomb rulers</th>
<th># consts</th>
<th>fault injected</th>
<th>non-conformity detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>2</td>
<td>p1ct2</td>
<td>[0 1 3 6]</td>
<td></td>
</tr>
<tr>
<td>CPUT2</td>
<td>4</td>
<td>p2ct3</td>
<td>[0 1 6 11]</td>
<td></td>
</tr>
<tr>
<td>CPUT3</td>
<td>5</td>
<td>p3ct3</td>
<td>[0 5 6 11]</td>
<td></td>
</tr>
<tr>
<td>CPUT4</td>
<td>9</td>
<td>p4ct5</td>
<td>sol(P4) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT5</td>
<td>9</td>
<td>p5ct1</td>
<td>sol(P5) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT6</td>
<td>9</td>
<td>p6ct2</td>
<td>[0 1 6 11]</td>
<td></td>
</tr>
<tr>
<td>CPUT7</td>
<td>9</td>
<td>p7ct9</td>
<td>sol(P7) = ∅</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Localization</th>
<th>Golomb rulers</th>
<th>fault localized</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>(rev(P1), (m_ct2))</td>
<td>0.32s</td>
<td></td>
</tr>
<tr>
<td>CPUT2</td>
<td>(rev(P2), (m_ct2))</td>
<td>0.82s</td>
<td></td>
</tr>
<tr>
<td>CPUT3</td>
<td>(rev(P3), (m_ct2))</td>
<td>1.15s</td>
<td></td>
</tr>
<tr>
<td>CPUT4</td>
<td>(rev(p4_ct5), ∅)</td>
<td>1.87s</td>
<td></td>
</tr>
<tr>
<td>CPUT5</td>
<td>(rev(p5_ct1), ∅)</td>
<td>1.62s</td>
<td></td>
</tr>
<tr>
<td>CPUT6</td>
<td>(rev(P6), (m_ct2))</td>
<td>2.14s</td>
<td></td>
</tr>
<tr>
<td>CPUT7</td>
<td>(rev(p7_ct6, p7_ct9), ∅)</td>
<td>1.92s</td>
<td></td>
</tr>
</tbody>
</table>
# Validation (Car Sequencing)

<table>
<thead>
<tr>
<th>car sequencing</th>
<th># constraints</th>
<th>fault injected</th>
<th>non-conformity detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>2</td>
<td>p1_ct2</td>
<td>slots1</td>
</tr>
<tr>
<td>CPUT2</td>
<td>3</td>
<td>p2_ct3</td>
<td>slots2</td>
</tr>
<tr>
<td>CPUT3</td>
<td>3</td>
<td>p3_ct2</td>
<td>slots3</td>
</tr>
<tr>
<td>CPUT4</td>
<td>3</td>
<td>p4_ct2</td>
<td>sol(P4) = Ø</td>
</tr>
<tr>
<td>CPUT5</td>
<td>6</td>
<td>p5_ct1</td>
<td>sol(P5) = Ø</td>
</tr>
<tr>
<td>CPUT6</td>
<td>6</td>
<td>p6_ct6</td>
<td>sol(P6) = Ø</td>
</tr>
<tr>
<td>CPUT7</td>
<td>6</td>
<td>p7_ct5</td>
<td>sol(P7) = Ø</td>
</tr>
</tbody>
</table>

slots1 = [4 5 3 6 4 6 5 1 3 2]
slots2 = [4 6 3 1 5 2 3 5 4 6]
slots3 = [5 2 3 6 1 4 3 6 4 5]

<table>
<thead>
<tr>
<th>cSeq</th>
<th>fault localized</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>(rev(P1), (m_ct2))</td>
<td>0.46s</td>
</tr>
<tr>
<td>CPUT2</td>
<td>(rev(P2), (m_ct2))</td>
<td>0.23s</td>
</tr>
<tr>
<td>CPUT3</td>
<td>(rev(P3), (m_ct2))</td>
<td>0.67s</td>
</tr>
<tr>
<td>CPUT4</td>
<td>(rev(p4_ct2), Ø)</td>
<td>0.87s</td>
</tr>
<tr>
<td>CPUT5</td>
<td>(rev(p5_ct1), Ø)</td>
<td>1.29s</td>
</tr>
<tr>
<td>CPUT6</td>
<td>(rev(p6_ct6), Ø)</td>
<td>1.34s</td>
</tr>
<tr>
<td>CPUT7</td>
<td>(rev(p7_ct5, p7_ct6), Ø)</td>
<td>1.28s</td>
</tr>
</tbody>
</table>

Intel Core 2 Duo CPU, 2.39 GHz, 2.00 Go RAM
## Validation (Car Sequencing)

<table>
<thead>
<tr>
<th>Test</th>
<th>car sequencing</th>
<th># constraints</th>
<th>fault injected</th>
<th>non-conformity detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>2</td>
<td>p1_ct2</td>
<td>slots1</td>
<td></td>
</tr>
<tr>
<td>CPUT2</td>
<td>3</td>
<td>p2_ct3</td>
<td>slots2</td>
<td></td>
</tr>
<tr>
<td><strong>CPUT3</strong></td>
<td><strong>3</strong></td>
<td><strong>p3_ct2</strong></td>
<td><strong>slots3</strong></td>
<td></td>
</tr>
<tr>
<td>CPUT4</td>
<td>3</td>
<td>p4_ct2</td>
<td>sol(P4) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT5</td>
<td>6</td>
<td>p5_ct1</td>
<td>sol(P5) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT6</td>
<td>6</td>
<td>p6_ct6</td>
<td>sol(P6) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT7</td>
<td>6</td>
<td>p7_ct5</td>
<td>sol(P7) = ∅</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Validation (Car Sequencing)</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>slots1 = [4 5 3 6 4 6 5 1 3 2]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>slots2 = [4 6 3 1 5 2 3 5 4 6]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>slots3 = [5 2 3 6 1 4 3 6 4 5]</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Localization</th>
<th>cSeq</th>
<th>fault localized</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>(rev(P1), (m_ct2))</td>
<td>0.46s</td>
<td></td>
</tr>
<tr>
<td>CPUT2</td>
<td>(rev(P2), (m_ct2))</td>
<td>0.23s</td>
<td></td>
</tr>
<tr>
<td><strong>CPUT3</strong></td>
<td>(rev(P3), (m_ct2))</td>
<td><strong>0.67s</strong></td>
<td></td>
</tr>
<tr>
<td>CPUT4</td>
<td>(rev(p4_ct2), ∅)</td>
<td>0.87s</td>
<td></td>
</tr>
<tr>
<td>CPUT5</td>
<td>(rev(p5_ct1), ∅)</td>
<td>1.29s</td>
<td></td>
</tr>
<tr>
<td>CPUT6</td>
<td>(rev(p6_ct6), ∅)</td>
<td>1.34s</td>
<td></td>
</tr>
<tr>
<td>CPUT7</td>
<td>(rev(p7_ct5, p7_ct6), ∅)</td>
<td>1.28s</td>
<td></td>
</tr>
</tbody>
</table>

*Intel Core 2 Duo CPU, 2.39 GHz, 2.00 Go RAM*
# Validation (Car Sequencing)

<table>
<thead>
<tr>
<th>car sequencing</th>
<th># constraints</th>
<th>fault injected</th>
<th>non-conformity detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>2</td>
<td>p1_ct2</td>
<td>slots1</td>
</tr>
<tr>
<td>CPUT2</td>
<td>3</td>
<td>p2_ct3</td>
<td>slots2</td>
</tr>
<tr>
<td>CPUT3</td>
<td>3</td>
<td>p3_ct2</td>
<td>slots3</td>
</tr>
<tr>
<td>CPUT4</td>
<td>3</td>
<td>p4_ct2</td>
<td>sol(P4) = ∅</td>
</tr>
<tr>
<td><strong>CPUT5</strong></td>
<td><strong>6</strong></td>
<td><strong>p5_ct1</strong></td>
<td><strong>sol(P5) = ∅</strong></td>
</tr>
<tr>
<td>CPUT6</td>
<td>6</td>
<td>p6_ct6</td>
<td>sol(P6) = ∅</td>
</tr>
<tr>
<td>CPUT7</td>
<td>6</td>
<td>p7_ct5</td>
<td>sol(P7) = ∅</td>
</tr>
</tbody>
</table>

slots1 = [4 5 3 6 4 6 5 1 3 2]
slots2 = [4 6 3 1 5 2 3 5 4 6]
slots3 = [5 2 3 6 1 4 3 6 4 5]

<table>
<thead>
<tr>
<th>cSeq</th>
<th>fault localized</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>(rev(P1), (m_ct2))</td>
<td>0.46s</td>
</tr>
<tr>
<td>CPUT2</td>
<td>(rev(P2), (m_ct2))</td>
<td>0.23s</td>
</tr>
<tr>
<td>CPUT3</td>
<td>(rev(P3), (m_ct2))</td>
<td>0.67s</td>
</tr>
<tr>
<td>CPUT4</td>
<td>(rev(p4_ct2), ∅)</td>
<td>0.87s</td>
</tr>
<tr>
<td><strong>CPUT5</strong></td>
<td><strong>(rev(p5_ct1), ∅)</strong></td>
<td><strong>1.29s</strong></td>
</tr>
<tr>
<td>CPUT6</td>
<td>(rev(p6_ct6), ∅)</td>
<td>1.34s</td>
</tr>
<tr>
<td>CPUT7</td>
<td>(rev(p7_ct5, p7_ct6), ∅)</td>
<td>1.28s</td>
</tr>
</tbody>
</table>

Intel Core 2 Duo CPU, 2.39 GHz, 2.00 Go RAM
# Validation (Car Sequencing)

<table>
<thead>
<tr>
<th>car sequencing</th>
<th># constraints</th>
<th>fault injected</th>
<th>non-conformity detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>2</td>
<td>p1_ct2</td>
<td>slots1</td>
</tr>
<tr>
<td>CPUT2</td>
<td>3</td>
<td>p2_ct3</td>
<td>slots2</td>
</tr>
<tr>
<td>CPUT3</td>
<td>3</td>
<td>p3_ct2</td>
<td>slots3</td>
</tr>
<tr>
<td>CPUT4</td>
<td>3</td>
<td>p4_ct2</td>
<td>sol(P4) = ∅</td>
</tr>
<tr>
<td>CPUT5</td>
<td>6</td>
<td>p5_ct1</td>
<td>sol(P5) = ∅</td>
</tr>
<tr>
<td>CPUT6</td>
<td>6</td>
<td>p6_ct6</td>
<td>sol(P6) = ∅</td>
</tr>
<tr>
<td>CPUT7</td>
<td>6</td>
<td>p7_ct5</td>
<td>sol(P7) = ∅</td>
</tr>
</tbody>
</table>

slots1 = [4 5 3 6 4 6 5 1 3 2]
slots2 = [4 6 3 1 5 2 3 5 4 6]
slots3 = [5 2 3 6 1 4 3 6 4 5]

### Localization

<table>
<thead>
<tr>
<th>cSeq</th>
<th>fault localized</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>(rev(P1), (m_ct2))</td>
<td>0.46s</td>
</tr>
<tr>
<td>CPUT2</td>
<td>(rev(P2), (m_ct2))</td>
<td>0.23s</td>
</tr>
<tr>
<td>CPUT3</td>
<td>(rev(P3), (m_ct2))</td>
<td>0.67s</td>
</tr>
<tr>
<td>CPUT4</td>
<td>(rev(p4_ct2), ∅)</td>
<td>0.87s</td>
</tr>
<tr>
<td>CPUT5</td>
<td>(rev(p5_ct1), ∅)</td>
<td>1.29s</td>
</tr>
<tr>
<td>CPUT6</td>
<td>(rev(p6_ct6), ∅)</td>
<td>1.34s</td>
</tr>
<tr>
<td>CPUT7</td>
<td>(rev(p7_ct5, p7_ct6), ∅)</td>
<td>1.28s</td>
</tr>
</tbody>
</table>
## Validation (n-queens)

<table>
<thead>
<tr>
<th>Test</th>
<th>8-queens</th>
<th># consts.</th>
<th>fault injected</th>
<th>non-conformity detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>p1_ct11</td>
<td>12</td>
<td></td>
<td>sol(P1) = Ø</td>
</tr>
<tr>
<td>CPUT2</td>
<td>p2_ct12</td>
<td>12</td>
<td></td>
<td>sol(P2) = Ø</td>
</tr>
<tr>
<td>CPUT3</td>
<td>p3_ct11</td>
<td>12</td>
<td></td>
<td>sol(P3) = Ø</td>
</tr>
<tr>
<td>CPUT4</td>
<td>p4_ct12</td>
<td>12</td>
<td></td>
<td>sol(P4) = Ø</td>
</tr>
<tr>
<td>CPUT5</td>
<td>p5_ct3</td>
<td>5</td>
<td></td>
<td>q1</td>
</tr>
<tr>
<td>CPUT6</td>
<td>p6_ct3</td>
<td>3</td>
<td></td>
<td>q2</td>
</tr>
<tr>
<td>CPUT7</td>
<td>p7_ct1</td>
<td>2</td>
<td></td>
<td>q3</td>
</tr>
</tbody>
</table>

### non-conformity
- q1 = [1 5 1 1 4 8 8 4]
- q2 = [7 8 3 2 4 1 5 6]
- q3 = [8 4 3 6 5 7 2 1]

<table>
<thead>
<tr>
<th>Localization</th>
<th>8-queens</th>
<th>fault localized</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>(rev(p1_ct11), Ø)</td>
<td>2.09s</td>
<td></td>
</tr>
<tr>
<td>CPUT2</td>
<td>(rev(p2_ct12), Ø)</td>
<td>1.64s</td>
<td></td>
</tr>
<tr>
<td>CPUT3</td>
<td>(rev(p3_ct11), Ø)</td>
<td>1.57s</td>
<td></td>
</tr>
<tr>
<td>CPUT4</td>
<td>(rev(p4_ct12), Ø)</td>
<td>1.59s</td>
<td></td>
</tr>
<tr>
<td>CPUT5</td>
<td>(rev(p5_ct3), Ø)</td>
<td>0.62s</td>
<td></td>
</tr>
<tr>
<td>CPUT6</td>
<td>(rev(P6), (m_ct2,m_ct3))</td>
<td>4.12s</td>
<td></td>
</tr>
<tr>
<td>CPUT7</td>
<td>(rev(P7), (m_ct2,m_ct3))</td>
<td>3.01s</td>
<td></td>
</tr>
</tbody>
</table>

Intel Core 2 Duo CPU, 2.39 GHz, 2.00 Go RAM
## Validation (n-queens)

<table>
<thead>
<tr>
<th>Test</th>
<th>8-queens</th>
<th># consts.</th>
<th>fault injected</th>
<th>non-conformity detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>12</td>
<td>p1_ct11</td>
<td>sol(P1) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT2</td>
<td>12</td>
<td>p2_ct12</td>
<td>sol(P2) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT3</td>
<td>12</td>
<td>p3_ct11</td>
<td>sol(P3) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT4</td>
<td>12</td>
<td>p4_ct12</td>
<td>sol(P4) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT5</td>
<td>5</td>
<td>p5_ct3</td>
<td>q1</td>
<td></td>
</tr>
<tr>
<td>CPUT6</td>
<td>3</td>
<td>p6_ct3</td>
<td>q2</td>
<td></td>
</tr>
<tr>
<td>CPUT7</td>
<td>2</td>
<td>p7_ct1</td>
<td>q3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Localization</th>
<th>8-queens</th>
<th>fault localized</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>(rev(p1_ct11), ∅)</td>
<td>2.09s</td>
<td></td>
</tr>
<tr>
<td>CPUT2</td>
<td>(rev(p2_ct12), ∅)</td>
<td>1.64s</td>
<td></td>
</tr>
<tr>
<td>CPUT3</td>
<td>(rev(p3_ct11), ∅)</td>
<td>1.57s</td>
<td></td>
</tr>
<tr>
<td>CPUT4</td>
<td>(rev(p4_ct12), ∅)</td>
<td>1.59s</td>
<td></td>
</tr>
<tr>
<td>CPUT5</td>
<td>(rev(p5_ct3), ∅)</td>
<td>0.62s</td>
<td></td>
</tr>
<tr>
<td>CPUT6</td>
<td>(rev(P6), (m_ct2, m_ct3))</td>
<td>4.12s</td>
<td></td>
</tr>
<tr>
<td>CPUT7</td>
<td>(rev(P7), (m_ct2, m_ct3))</td>
<td>3.01s</td>
<td></td>
</tr>
</tbody>
</table>

Intel Core 2 Duo CPU, 2.39 GHz, 2.00 Go RAM
Validation (Social Golfer)

<table>
<thead>
<tr>
<th>Test</th>
<th>social golfer</th>
<th># constrs.</th>
<th>fault injected</th>
<th>non-conformity detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>5</td>
<td>p1_ct1</td>
<td>sol(P1) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT2</td>
<td>5</td>
<td>p2_ct2</td>
<td>ncl</td>
<td></td>
</tr>
<tr>
<td>CPUT3</td>
<td>5</td>
<td>p3_ct3</td>
<td>sol(P3) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT4</td>
<td>5</td>
<td>p4_ct4</td>
<td>sol(P4) = ∅</td>
<td></td>
</tr>
<tr>
<td>CPUT5</td>
<td>5</td>
<td>p5_ct5</td>
<td>sol(P5) = ∅</td>
<td></td>
</tr>
</tbody>
</table>

non-conformity
ncl = [[1 1 1 1][1 2 2 2][1 3 3 3]
[2 1 3 3][2 2 2 1][3 3 3 2][3 1 1 2][3 3 2 3]]

<table>
<thead>
<tr>
<th>Localization</th>
<th>social golfer</th>
<th>fault localized</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>(rev(p1_ct1), ∅)</td>
<td>1.04s</td>
<td></td>
</tr>
<tr>
<td>CPUT2</td>
<td>(rev(P2), (m_ct2))</td>
<td>0.18s</td>
<td></td>
</tr>
<tr>
<td>CPUT3</td>
<td>(rev(p3_ct3), ∅)</td>
<td>1.05s</td>
<td></td>
</tr>
<tr>
<td>CPUT4</td>
<td>(rev(p4_ct4, p4_ct5), ∅)</td>
<td>1.03s</td>
<td></td>
</tr>
<tr>
<td>CPUT5</td>
<td>(rev(p5_ct5), ∅)</td>
<td>1.15s</td>
<td></td>
</tr>
</tbody>
</table>
Validation (Social Golfer)

<table>
<thead>
<tr>
<th>Test</th>
<th>social golfer</th>
<th># constrs.</th>
<th>fault injected</th>
<th>non-conformity detected</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>5</td>
<td>p1_ct1</td>
<td></td>
<td>$\text{sol}(P1) = \emptyset$</td>
</tr>
<tr>
<td>CPUT2</td>
<td>5</td>
<td>p2_ct2</td>
<td></td>
<td>ncl1</td>
</tr>
<tr>
<td>CPUT3</td>
<td>5</td>
<td>p3_ct3</td>
<td></td>
<td>$\text{sol}(P3) = \emptyset$</td>
</tr>
<tr>
<td>CPUT4</td>
<td>5</td>
<td>p4_ct4</td>
<td></td>
<td>$\text{sol}(P4) = \emptyset$</td>
</tr>
<tr>
<td>CPUT5</td>
<td>5</td>
<td>p5_ct5</td>
<td></td>
<td>$\text{sol}(P5) = \emptyset$</td>
</tr>
</tbody>
</table>

| non-conformity | ncl= [1 1 1 1][1 2 2 2][1 3 3 3][2 1 3 3][2 2 2 1][2 2 1 1][3 3 3 2][3 1 1 2][3 3 2 3] |

<table>
<thead>
<tr>
<th>Localization</th>
<th>social golfer</th>
<th>fault localized</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUT1</td>
<td>(rev(p1_ct1),\emptyset)</td>
<td>1.04s</td>
<td></td>
</tr>
<tr>
<td>CPUT2</td>
<td>(rev(p2), (m_ct2))</td>
<td>0.18s</td>
<td></td>
</tr>
<tr>
<td>CPUT3</td>
<td>(rev(p3_ct3),\emptyset)</td>
<td>1.05s</td>
<td></td>
</tr>
<tr>
<td>CPUT4</td>
<td>(rev(p4_ct4,p4_ct5),\emptyset)</td>
<td>1.03s</td>
<td></td>
</tr>
<tr>
<td>CPUT5</td>
<td>(rev(p5_ct5),\emptyset)</td>
<td>1.15s</td>
<td></td>
</tr>
</tbody>
</table>
Conclusion

Fault localization in constraint programs is possible

But, our method is based on the hypothesis that there is a single fault in the program!
Conclusion

Fault localization in constraint programs is possible

But, our method is based on the hypothesis that there is a single fault in the program!

Perspectives

• Extend to multiple faults by considering $\text{Powset}\{C_1, \ldots, C_n\}$

• Extend to better correction by calculating the correction set
Conclusion

Fault localization in constraint programs is possible

But, our method is based on the hypothesis that there is a single fault in the program!

Perspectives

• Extend to multiple faults by considering $\text{Powset}({C_1, \ldots C_n})$

• Extend to better correction by calculating the correction set
Existing debugging tools

- **Syntax checking**
  (OPL studio [IBM ILOG]…)

- **Host’s language debuggers**
  (gdb for Gecode, jdb for Choco…)

- **Post-mortem trace analyzers**
  (Codeine for Prolog[Langevine03], Morphine for Mercury,[Jahier02], ILOG Gentra4CP[Deransart 04], [Jussien96] Jpalm/Jchoco …)

- **Trace visualization**
  (CLPGUI[Fages04], CP-VIZ[Simonis10] …)