DFA-based formulation for constraint negation

Nadjib Lazaar, Nourredine Aribi, Arnaud Gotlieb, Yahia Lebbah

ModRef Workshop
8 Oct. 2012
Motivations

- Many CP-application domains are coming to challenge modern CP languages and their global constraints tool-box:
  - Program verification [Collavizza et al., 10][Lazaar et al., 10]
  - Reinforcement learning [Bessière et al., 07]
Motivations

• Many CP-application domains are coming to challenge modern CP languages and their global constraints tool-box:
  o Program verification [Collavizza et al.,10][Lazaar et al.,10]
  o Reinforcement learning [Bessière et al.,07]

The need: Global constraint negation!
Motivations

• Many CP-application domains are coming to challenge modern CP languages and their global constraints tool-box:
  o Program verification [Collavizza et al.,10][Lazaar et al.,10]
  o Reinforcement learning [Bessière et al.,07]

The need: Global constraint negation!

How:
  o Reformulation (e.g., negation of an \texttt{atLeast} is an \texttt{atMost})
  o Decomposition and syntactic transformation:
Motivations

- Many CP-application domains are coming to challenge modern CP languages and their global constraints tool-box:
  - Program verification \cite{Collavizza10,Lazaar10}
  - Reinforcement learning \cite{Bessiere07}

The need: Global constraint negation!

How:
- Reformulation (e.g., negation of an \texttt{atLeast} is an \texttt{atMost})
- Decomposition and syntactic transformation:

\textbf{INVERSE}(tab_1, tab_2)

\begin{align*}
\forall i \in D_1 : \quad & tab_1[tab_2[i]] = i \\
\forall j \in D_2 : \quad & tab_2[tab_1[j]] = j
\end{align*}
Motivations

• Many CP-application domains are coming to challenge modern CP languages and their global constraints tool-box:
  o Program verification [Collavizza et al.,10][Lazaar et al.,10]
  o Reinforcement learning [Bessière et al.,07]

The need: Global constraint negation!

How:
  o Reformulation (e.g., negation of an \texttt{atLeast} is an \texttt{atMost})
  o Decomposition and syntactic transformation:
Motivations

• More and more generic constraint representations and reformulations are proposed:
  o Deterministic finite automaton (DFA) [Pesant,04]
  o Multivalued Decision Diagram (MDD) [Andersen et al.,07][Hoda et al.,10]
  o Grammar [Quimper et al., 06][Katsirelos et al.,09]
  o Berge-acyclic graph [Beldiceanu et al., catalog11]
  o ...
Motivations

• More and more generic constraint representations and reformulations are proposed:
  o Deterministic finite automaton (DFA) [Pesant,04]
  o Multivalued Decision Diagram (MDD) [Andersen et al.,07][Hoda et al.,10]
  o Grammar [Quimper et al., 06][Katsirelos et al.,09]
  o Berge-acyclic graph [Beldiceanu et al., catalog11]
  o ...

The need
  o Generic global constraint negation
Motivations

• More and more generic constraint representations and reformulations are proposed:
  o **Deterministic finite automaton (DFA)** [Pesant,04]
  o Multivalued Decision Diagram (MDD) [Andersen et al.,07][Hoda et al.,10]
  o Grammar [Quimper et al., 06][Katsirelos et al.,09]
  o Berge-acyclic graph [Beldiceanu et al., catalog11]
  o ...

The need
  o Generic global constraint negation

This paper
  o **DFA-based global constraint negation**
**Regular constraint**

- DFAs accept precisely regular languages

\[
\text{REGULAR}(x_1 \ldots x_n, A) \text{ holds iff:}
\]

  - \( x_1 \ldots x_n \) is a string accepted by the deterministic finite automaton \( A \)

- Many global constraints are instances of \text{Regular}
  
  - Among, Contiguity, Lex, Precedence …

- 3-stages for the consistency algorithm (\( O(nd|E|) \))
  
  - Forward step
  - Backward step
  - Maintaining step
Negation on DFA-based GC

Let $\mathcal{A} = (X, E, \Psi, \Sigma, \delta, e_0, F)$ a DFA of a given constraint $C$:

- $X$ a sequence of finite-domain variables
- $E$ a finite set of states $e_i$
- $\Psi$ a finite set of labeled states: (source, node, sink and final states)
- $\Sigma$ a finite alphabet
- $\delta$ a transition function
- $e_0$ the start state
- $F$ a subset of final states
Negation on DFA-based GC

Property [hopcroft et al.,06]: The complement of a regular language is regular
Negation on DFA-based GC

Property [hopcroft et al.,06]: The complement of a regular language is regular

Proposition: The complement of a DFA of a given constraint $C$ represents the DFA of the negated form $\text{not}C$
Negation on DFA-based GC

Property [Hopcroft et al., 06]: The **complement** of a regular language is regular.

Proposition: The **complement** of a DFA of a given constraint $C$ represents the DFA of the negated form $\text{not}C$. 

![DFA of C]

---

DFA of $C$:

- States: $e_0, e_1, e_2, e_3$
- Alphabet: $\{0, 1\}$
- Transitions:
  - $e_0 \rightarrow e_0$ on $0$
  - $e_0 \rightarrow e_1$ on $1$
  - $e_1 \rightarrow e_2$ on $1$
  - $e_1 \rightarrow e_3$ on $0$
  - $e_2 \rightarrow e_3$ on $0$
  - $e_3 \rightarrow e_3$ on $1$
Property \cite{hopcroft_et_al_06}: The complement of a regular language is regular.

Proposition: The complement of a DFA of a given constraint $C$ represents the DFA of the negated form $\neg C$.
We get the \textbf{complement} of a global constraint DFA’s
By combining the 3 following operators:
Negation on DFA-based GC

We get the complement of a global constraint DFA’s. By combining the 3 following operators:

- **Complete automaton operator** $C(A)$

The complete automaton of $A = (X, E, \Psi, \Sigma, \delta, e_0, F)$ is $C(A) = (X, E', \Psi', \Sigma, \delta', e_0, F)$ s.t.:

- $E' = E \cup \{e_k : \exists s \in \Sigma, e_i \in E \text{ s.t. } (e_i, s, e_k) \notin \delta\}$
- $\Psi' = \Psi \cup \{sink(e_k) : e_k \in E' \setminus E\}$
- $\delta' = \delta \cup \{(e_i, s, e_k) : \exists s \in \Sigma, e_i \in E', e_k \notin E\}$
Negation on DFA-based GC

We get the **complement** of a global constraint DFA’s By combining the 3 following operators:

- Complete automaton operator $C(A)$
- Swap-state operator $S(A)$

A swap state on $A = (X, E, \Psi, \Sigma, \delta, e_0, F)$ is $S(A) = (X, E, \Psi', \Sigma, \delta, e_0, F')$ s.t.:

$$\forall e_i, e_j \in E : \text{final}(e_i), \text{sink}(e_j) \in \Psi \Rightarrow \text{sink}(e_i), \text{final}(e_j) \in \Psi'$$
Negation on DFA-based GC

We get the **complement** of a global constraint DFA’s by combining the 3 following operators:

- **Complete automaton operator** \( C(A) \)
- **Swap-state operator** \( S(A) \)
- **Clean-up operator** \( U(A) \)

A clean-up on \( A = (X, E, \Psi, \Sigma, \delta, e_0, F) \) is \( U(A) = (X, E', \Psi', \Sigma, \delta', e_0, F) \) s.t.:

\[
- E' = E \setminus \{ e_k : \text{sink}(e_k) \in \Psi \} \\
- \Psi' = \Psi \setminus \{ \text{sink}(e_k) : e_k \in E \} \\
- \delta' = \delta \setminus \{ (e_i, s, e_k) \in E \times \Sigma \times E : \text{sink}(e_k) \in \Psi \}
\]
Negation on DFA-based GC

We get the **complement** of a global constraint DFA’s by combining the 3 following operators:

- Complete automaton operator $C(A)$
- Swap-state operator $S(A)$
- Clean-up operator $U(A)$

$\overline{A}$ is the complement of $A$ s.t.:

\[
\overline{A} = U(S(C(A)))
\]
Example (Global-contiguity)

- **Global-contiguity**\((x_1...x_n)\), This constraint enforce all variables \(x_1...x_n\) to be assigned value 0 or 1, and all variables assigned to value 1 appear contiguously.

**Global-contiguity DFA:**

![Diagram](image_url)
Example (Global_contiguity)

- **Global-contiguity**($x_1 ... x_n$), This constraint enforces all variables $x_1 ... x_n$ to be assigned value 0 or 1, and all variables assigned to value 1 appear contiguously.

1- Complete:
Example (Global-contiguity)

- **Global-contiguity**\((x_1 \ldots x_n)\), This constraint enforce all variables \(x_1 \ldots x_n\) to be assigned value 0 or 1, and all variables assigned to value 1 appear contiguously.

2- Swap-state:
Example (Global-contiguity)

- \textbf{Global-contiguity}(x_1 \ldots x_n), \ This \ constraint \ enforce \ all \ variables \ x_1 \ldots x_n \ to \ be \ assigned \ value \ 0 \ or \ 1, \ and \ all \ variables \ assigned \ to \ value \ 1 \ appear \ contiguously.\n
3- Clean-up:
Example (\textsc{Global-contiguity})

- \textsc{Global-contiguity}(x_1 \ldots x_n), This constraint enforces all variables \(x_1 \ldots x_n\) to be assigned values 0 or 1, and all variables assigned to value 1 appear contiguously.

3- Clean-up:

DFA of \texttt{not(Global-contiguity)}
Example (Global_contiguity)

- Filtering using **regular**: 
  - Let $x_1$ to $x_4$ sequence of variables in $\{0,1\}$

\[
\begin{align*}
X_1 & \{0,1\} \\
X_2 & \{0,1\} \\
X_3 & \{0,1\} \\
X_4 & \{0,1\}
\end{align*}
\]
Example (Global_contiguity)

- Filtering using **regular**:
  - Let $x_1$ to $x_4$ sequence of variables in \{0,1\}

\[
\begin{align*}
X_1 & \in \{0,1\} \\
X_2 & \in \{0,1\} \\
X_3 & \in \{0,1\} \\
X_4 & \in \{0,1\}
\end{align*}
\]
Example (Global_contiguity)

- Filtering using **regular**:
  - Let $x_1$ to $x_4$ sequence of variables in $\{0, 1\}$

```
X_1
{0,1}

X_2
{0,1}

X_3
{0,1}

X_4
{0,1}
```
Example (Global_contiguity)

- Filtering using **Regular**:
  - Let $x_1$ to $x_4$ sequence of variables in \{0,1\}

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>{1}</td>
<td>{0}</td>
<td>{1}</td>
<td>{0,1}</td>
</tr>
</tbody>
</table>
Experimental validation

- **Global_contiguity**

Decomposition of **Global_contiguity** in primitive constraints gives:

\[
global\_contiguity(x) \equiv \forall i, j \in 1..n : i < j \text{ s.t.} \quad (x_i = 1) \land (x_j = 0) \Rightarrow (\forall k \in j + 1..n : x_k = 0)
\]

Syntactic negation:

\[
\neg global\_contiguity(x) \equiv \exists i, j \in 1..n : i < j \text{ s.t.} \quad (x_i = 1) \land (x_j = 0) \land (\exists k \in j + 1..n : x_k = 1)
\]
Experimental validation

- **Global_contiguity**

<table>
<thead>
<tr>
<th>var</th>
<th>syntactic transformations based negation</th>
<th>DFA – based negation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td>M</td>
</tr>
<tr>
<td>200</td>
<td>53.13</td>
<td>24.39</td>
</tr>
<tr>
<td>300</td>
<td>186.07</td>
<td>77.90</td>
</tr>
<tr>
<td>400</td>
<td>509.93</td>
<td>179.22</td>
</tr>
<tr>
<td>500</td>
<td>1 082.35</td>
<td>344.86</td>
</tr>
<tr>
<td>600</td>
<td>1 936.68</td>
<td>589.79</td>
</tr>
<tr>
<td>700</td>
<td>3 125.01</td>
<td>930.34</td>
</tr>
<tr>
<td>800</td>
<td>4 735.71</td>
<td>1 381.68</td>
</tr>
<tr>
<td>900</td>
<td>6 760.44</td>
<td>1 960.24</td>
</tr>
<tr>
<td>1 000</td>
<td>19 407.02</td>
<td>2 681.01</td>
</tr>
<tr>
<td>1 100</td>
<td>—</td>
<td>OOM</td>
</tr>
<tr>
<td>1 200</td>
<td>—</td>
<td>OOM</td>
</tr>
<tr>
<td>1 300</td>
<td>—</td>
<td>OOM</td>
</tr>
<tr>
<td>1 400</td>
<td>—</td>
<td>OOM</td>
</tr>
<tr>
<td>1 500</td>
<td>—</td>
<td>OOM</td>
</tr>
<tr>
<td>1 600</td>
<td>—</td>
<td>OOM</td>
</tr>
<tr>
<td>1 700</td>
<td>—</td>
<td>OOM</td>
</tr>
<tr>
<td>1 800</td>
<td>—</td>
<td>OOM</td>
</tr>
<tr>
<td>1 900</td>
<td>—</td>
<td>OOM</td>
</tr>
<tr>
<td>11 000</td>
<td>—</td>
<td>OOM</td>
</tr>
</tbody>
</table>

T: time(ms), M: memory(MB), P: propagations, N: nodes, OOM: Out Of Memory

Intel Core2Duo CPU, Q6600 of 2.4Ghz with 3Gb of RAM
Experimental validation

- **Global_contiguity**

<table>
<thead>
<tr>
<th>var</th>
<th>syntactic transformations based negation</th>
<th>DFA - based negation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td>M</td>
</tr>
<tr>
<td>200</td>
<td>53.13</td>
<td>24.39</td>
</tr>
<tr>
<td>300</td>
<td>186.07</td>
<td>77.90</td>
</tr>
<tr>
<td>400</td>
<td>509.93</td>
<td>179.22</td>
</tr>
<tr>
<td>500</td>
<td>1,082.35</td>
<td>344.86</td>
</tr>
<tr>
<td>600</td>
<td>1,936.68</td>
<td>589.79</td>
</tr>
<tr>
<td>700</td>
<td>3,125.01</td>
<td>930.34</td>
</tr>
<tr>
<td>800</td>
<td>4,735.71</td>
<td>1,381.68</td>
</tr>
<tr>
<td>900</td>
<td>6,760.44</td>
<td>1,960.24</td>
</tr>
<tr>
<td>1,000</td>
<td>19,407.02</td>
<td>2,681.01</td>
</tr>
<tr>
<td>1,100</td>
<td>—</td>
<td>ODM</td>
</tr>
<tr>
<td>1,200</td>
<td>—</td>
<td>ODM</td>
</tr>
<tr>
<td>1,300</td>
<td>—</td>
<td>ODM</td>
</tr>
<tr>
<td>1,400</td>
<td>—</td>
<td>ODM</td>
</tr>
<tr>
<td>1,500</td>
<td>—</td>
<td>ODM</td>
</tr>
<tr>
<td>1,600</td>
<td>—</td>
<td>ODM</td>
</tr>
<tr>
<td>1,700</td>
<td>—</td>
<td>ODM</td>
</tr>
<tr>
<td>1,800</td>
<td>—</td>
<td>ODM</td>
</tr>
<tr>
<td>1,900</td>
<td>—</td>
<td>ODM</td>
</tr>
<tr>
<td>2,000</td>
<td>—</td>
<td>ODM</td>
</tr>
</tbody>
</table>

Intel Core2Duo CPU, Q6600 of 2.4Ghz with 3Gb of RAM
Experimental validation

\[ \leq_{\text{LEX}} \]

Decomposition of \( \leq_{\text{LEX}} \) in primitive constraints gives:

\[
x \leq_{\text{LEX}} y \equiv (n = 0) \lor (x_0 < y_0) \lor
\]
\[
(x_0 = y_0 \land < x_1, \ldots, x_{n-1} > \leq_{\text{LEX}} < y_1, \ldots, y_{n-1} >)
\]

Syntactic negation:

\[
\neg (x \leq_{\text{LEX}} y) \equiv ((n = 1) \land (x_0 > y_0)) \lor ((n > 1) \land ((x_0 > y_0) \lor
\]
\[
(< x_1, \ldots, x_{n-1} > \neg \leq_{\text{LEX}} < y_1, \ldots, y_{n-1} >))
\]

with a reformulation-based negation:

\[
\neg (x \leq_{\text{LEX}} y) \equiv x >_{\text{LEX}} y
\]
Experimental validation

- \( \leq_{\text{lex}} \)

<table>
<thead>
<tr>
<th>var</th>
<th>syntactic transformations</th>
<th>( \geq_{\text{lex}} )</th>
<th>DFA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>T</td>
<td>M</td>
<td>P</td>
</tr>
<tr>
<td>200</td>
<td>7.00</td>
<td>1.67</td>
<td>1 944</td>
</tr>
<tr>
<td>300</td>
<td>13.24</td>
<td>3.50</td>
<td>2 708</td>
</tr>
<tr>
<td>400</td>
<td>21.45</td>
<td>6.42</td>
<td>3 544</td>
</tr>
<tr>
<td>500</td>
<td>30.86</td>
<td>9.56</td>
<td>4 308</td>
</tr>
<tr>
<td>600</td>
<td>43.32</td>
<td>13.35</td>
<td>5 144</td>
</tr>
<tr>
<td>700</td>
<td>56.77</td>
<td>17.38</td>
<td>5 908</td>
</tr>
<tr>
<td>800</td>
<td>71.89</td>
<td>22.00</td>
<td>6 744</td>
</tr>
<tr>
<td>900</td>
<td>90.12</td>
<td>27.12</td>
<td>7 508</td>
</tr>
<tr>
<td>(10^3)</td>
<td>107.97</td>
<td>33.02</td>
<td>8 344</td>
</tr>
<tr>
<td>2.10(^3)</td>
<td>402.09</td>
<td>123.06</td>
<td>16 344</td>
</tr>
<tr>
<td>3.10(^3)</td>
<td>889.68</td>
<td>270.51</td>
<td>24 344</td>
</tr>
<tr>
<td>4.10(^3)</td>
<td>1 591.25</td>
<td>475.89</td>
<td>32 344</td>
</tr>
<tr>
<td>5.10(^3)</td>
<td>2 527.08</td>
<td>738.53</td>
<td>40 344</td>
</tr>
<tr>
<td>6.10(^3)</td>
<td>3 758.49</td>
<td>1 059.63</td>
<td>48 344</td>
</tr>
<tr>
<td>7.10(^3)</td>
<td>5 194.56</td>
<td>1 440.67</td>
<td>56 344</td>
</tr>
<tr>
<td>8.10(^3)</td>
<td>6 951.67</td>
<td>1 880.27</td>
<td>64 344</td>
</tr>
</tbody>
</table>

T: time(ms), M: memory(MB), P: propagations, N: nodes, OOM: Out Of Memory

Intel Core2Duo CPU, Q6600 of 2.4Ghz with 3Gb of RAM
Experimental validation

- $< \text{lex}$

<table>
<thead>
<tr>
<th>var</th>
<th>syntactic transformations</th>
<th>$\text{lex}$</th>
<th>DFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>M</td>
<td>P</td>
<td>N</td>
</tr>
<tr>
<td>200</td>
<td>7.00</td>
<td>1.67</td>
<td>1944</td>
</tr>
<tr>
<td>300</td>
<td>13.24</td>
<td>3.45</td>
<td>2708</td>
</tr>
<tr>
<td>400</td>
<td>21.45</td>
<td>6.42</td>
<td>3544</td>
</tr>
<tr>
<td>500</td>
<td>30.86</td>
<td>9.56</td>
<td>4308</td>
</tr>
<tr>
<td>600</td>
<td>43.32</td>
<td>13.35</td>
<td>5144</td>
</tr>
<tr>
<td>700</td>
<td>56.77</td>
<td>17.38</td>
<td>5908</td>
</tr>
<tr>
<td>800</td>
<td>71.89</td>
<td>22.00</td>
<td>6744</td>
</tr>
<tr>
<td>900</td>
<td>90.12</td>
<td>27.12</td>
<td>7508</td>
</tr>
<tr>
<td>$10^3$</td>
<td>107.97</td>
<td>33.02</td>
<td>8344</td>
</tr>
<tr>
<td>$2.10^3$</td>
<td>402.09</td>
<td>123.06</td>
<td>16344</td>
</tr>
<tr>
<td>$3.10^3$</td>
<td>889.68</td>
<td>270.51</td>
<td>24344</td>
</tr>
<tr>
<td>$4.10^3$</td>
<td>1591.25</td>
<td>475.89</td>
<td>32344</td>
</tr>
<tr>
<td>$5.10^3$</td>
<td>2527.08</td>
<td>738.53</td>
<td>40344</td>
</tr>
<tr>
<td>$6.10^3$</td>
<td>3758.49</td>
<td>1059.63</td>
<td>48344</td>
</tr>
<tr>
<td>$7.10^3$</td>
<td>5104.56</td>
<td>1410.67</td>
<td>56344</td>
</tr>
</tbody>
</table>

Intel Core2Duo CPU, Q6600 of 2.4Ghz with 3Gb of RAM
Conclusions and perspectives

• A preliminary approach to an automatic negation on DFA-based constraints
  o DFA operators (complete, swap-state, clean-up)
  o A Negation for free by exploiting \texttt{REGULAR} consistency algorithm
Conclusions and perspectives

• A preliminary approach to an automatic negation on DFA-based constraints
  o DFA operators (complete, swap-state, clean-up)
  o A Negation for free by exploiting \textsc{regular} consistency algorithm

• Extend the approach on other generic representation (MDD, \textsc{grammar},…)

Conclusions and perspectives

• A preliminary approach to an automatic negation on DFA-based constraints
  o DFA operators (complete, swap-state, clean-up)
  o A Negation for free by exploiting regular consistency algorithm
• Extend the approach on other generic representation (MDD, grammar, …)
• Generic global constraint combinaisons (i.e., conjunction, disjunction)
Conclusions and perspectives

- A preliminary approach to an automatic negation on DFA-based constraints
  - DFA operators (complete, swap-state, clean-up)
  - A Negation for free by exploiting regular consistency algorithm

- Extend the approach on other generic representation (MDD, grammar, …)

- Generic global constraint combinaisons (i.e., conjunction, disjunction)

Thank you