Efficient Inclusion for a Class of XML Types with Interleaving and Counting

D. Colazzo, G. Ghelli, C. Sartiani

1st Codex meeting

XML schemas

- Essential tool to check and enforce safety and correctness of XML applications.

XML schemas

- Essential tool to check and enforce safety and correctness of XML applications.
- Schema inclusion and schema equality are the main ingredients of any kind of static analysis.

XML schemas

- Essential tool to check and enforce safety and correctness of XML applications.
- Schema inclusion and schema equality are the main ingredients of any kind of static analysis.
- Schema inclusion is PSPACE (or EXPTIME for Extended DTDs) complete when only ordered types are adopted (like in DTDs)

XML schemas

- Essential tool to check and enforce safety and correctness of XML applications.
- Schema inclusion and schema equality are the main ingredients of any kind of static analysis.
- Schema inclusion is PSPACE (or EXPTIME for Extended DTDs) complete when only ordered types are adopted (like in DTDs)

Counting and interleaving

- Type mechanisms considered in Regular Expressions (REs) for XML Schema and Relax-NG

Counting and interleaving

- Type mechanisms considered in Regular Expressions (REs) for XML Schema and Relax-NG
- The only presence of interleaving makes RE inclusion EXPSPACE complete!

Counting and interleaving

- Type mechanisms considered in Regular Expressions (REs) for XML Schema and Relax-NG
- The only presence of interleaving makes RE inclusion EXPSPACE complete!
- Mayer and Stockmeyer showed that counting is not necessary for EXPSPACE completeness.

Counting and interleaving

- Type mechanisms considered in Regular Expressions (REs) for XML Schema and Relax-NG
- The only presence of interleaving makes RE inclusion EXPSPACE complete!
- Mayer and Stockmeyer showed that counting is not necessary for EXPSPACE completeness.
- A recent ICDT paper by Gelade, Martens, and Neven left an open problem "It would therefore be desirable to find *robust subclasses* for which the basic decision problems are in PTIME".

Counting and interleaving

- Type mechanisms considered in Regular Expressions (REs) for XML Schema and Relax-NG
- The only presence of interleaving makes RE inclusion EXPSPACE complete!
- Mayer and Stockmeyer showed that counting is not necessary for EXPSPACE completeness.
- A recent ICDT paper by Gelade, Martens, and Neven left an open problem "It would therefore be desirable to find *robust subclasses* for which the basic decision problems are in PTIME".
- This work: a first answer to this question.

Counting and interleaving

- Type mechanisms considered in Regular Expressions (REs) for XML Schema and Relax-NG
- The only presence of interleaving makes RE inclusion EXPSPACE complete!
- Mayer and Stockmeyer showed that counting is not necessary for EXPSPACE completeness.
- A recent ICDT paper by Gelade, Martens, and Neven left an open problem "It would therefore be desirable to find *robust subclasses* for which the basic decision problems are in PTIME".
- This work: a first answer to this question.

Our robust subclass : Conflict-Free types

- We focus on REs, and prove that inclusion between two *Conflict-Free* REs T and U can be checked polynomial time (more precisely in $O(|T| * |U| + |U|^2)$ time)

Our robust subclass : Conflict-Free types

- We focus on REs, and prove that inclusion between two *Conflict-Free* REs T and U can be checked polynomial time (more precisely in $O(|T| * |U| + |U|^2)$ time)
- The result (polynomial complexity) directly extends to C-F XML schemas thanks to a recent result by Gelade, Martens, and Neven :

Our robust subclass : Conflict-Free types

- We focus on REs, and prove that inclusion between two *Conflict-Free* REs T and U can be checked polynomial time (more precisely in $O(|T| * |U| + |U|^2)$ time)
- The result (polynomial complexity) directly extends to C-F XML schemas thanks to a recent result by Gelade, Martens, and Neven :
  - if inclusion for a given class of **REs** with interleaving and numerical constraints is in the complexity class $\mathcal{C}$,...

Our robust subclass : Conflict-Free types

- We focus on REs, and prove that inclusion between two *Conflict-Free* REs T and U can be checked polynomial time (more precisely in $O(|T| * |U| + |U|^2)$ time)
- The result (polynomial complexity) directly extends to C-F XML schemas thanks to a recent result by Gelade, Martens, and Neven :
  - if inclusion for a given class of **REs** with interleaving and numerical constraints is in the complexity class $\mathcal{C}$,...
  - ..and $\mathcal{C}$ is *closed under positive reductions* (a property enjoyed by PTIME),..

Our robust subclass : Conflict-Free types

- We focus on REs, and prove that inclusion between two *Conflict-Free* REs T and U can be checked polynomial time (more precisely in $O(|T| * |U| + |U|^2)$ time)
- The result (polynomial complexity) directly extends to C-F XML schemas thanks to a recent result by Gelade, Martens, and Neven :
  - if inclusion for a given class of **REs** with interleaving and numerical constraints is in the complexity class $\mathcal{C}$,...
  - ..and $\mathcal{C}$ is *closed under positive reductions* (a property enjoyed by PTIME),..
  - .. then the complexity of inclusion for DTDs and single-type EDTDs that use the same class of REs is in $\mathcal{C}$ too
- Single-type EDTDs are the theoretical counterpart of XML Schema definitions.

Our robust subclass : Conflict-Free types

- We focus on REs, and prove that inclusion between two *Conflict-Free* REs T and U can be checked polynomial time (more precisely in $O(|T| * |U| + |U|^2)$ time)
- The result (polynomial complexity) directly extends to C-F XML schemas thanks to a recent result by Gelade, Martens, and Neven :
  - if inclusion for a given class of **REs** with interleaving and numerical constraints is in the complexity class $\mathcal{C}$,...
  - ..and $\mathcal{C}$ is *closed under positive reductions* (a property enjoyed by PTIME),..
  - .. then the complexity of inclusion for DTDs and single-type EDTDs that use the same class of REs is in $\mathcal{C}$ too
- Single-type EDTDs are the theoretical counterpart of XML Schema definitions.

Conflict-Free REs, first restriction

$$T ::= \quad \epsilon \;\mid\; a\,[m..n] \;\mid\; T + T \;\mid\; T \cdot T \;\mid\; T \& T \;\mid\; T!$$

Conflict-Free REs, first restriction

$$T ::= \quad \epsilon \;\mid\; a[m..n] \;\mid\; T+T \;\mid\; T\cdot T \;\mid\; T\&T \;\mid\; T!$$

$$[\![\epsilon]\!] \quad = \quad \{\epsilon\}$$

Conflict-Free REs, first restriction

$$T ::= \quad \epsilon \;\mid\; a\,[m..n] \;\mid\; T + T \;\mid\; T \cdot T \;\mid\; T \& T \;\mid\; T!$$

$$\begin{array}{rcl} [\![\epsilon]\!] & = & \{\epsilon\} \\ [\![a\,[m..n]]\!] & = & \{w \mid S(w) = \{a\},\; m \leq |w| \leq n\} \end{array}$$

Conflict-Free REs, first restriction

$$T ::= \epsilon \mid a[m..n] \mid T + T \mid T \cdot T \mid T \& T \mid T!$$

$$\begin{array}{rcl}
[\![\epsilon]\!] & = & \{\epsilon\} \\
[\![a[m..n]]\!] & = & \{w \mid S(w) = \{a\},\ m \leq |w| \leq n\} \\
[\![T_1 + T_2]\!] & = & [\![T_1]\!] \cup [\![T_2]\!]
\end{array}$$

Conflict-Free REs, first restriction

$$T ::= \epsilon \mid a[m..n] \mid T + T \mid T \cdot T \mid T \& T \mid T!$$

$$\begin{array}{rcl}
[\![\epsilon]\!] & = & \{\epsilon\} \\
[\![a[m..n]]\!] & = & \{w \mid S(w) = \{a\},\ m \le |w| \le n\} \\
[\![T_1 + T_2]\!] & = & [\![T_1]\!] \cup [\![T_2]\!] \\
[\![T_1 \cdot T_2]\!] & = & [\![T_1]\!] \cdot [\![T_2]\!]
\end{array}$$

Conflict-Free REs, first restriction

$$T ::= \;\; \epsilon \;\mid\; a\,[m..n] \;\mid\; T + T \;\mid\; T \cdot T \;\mid\; T \& T \;\mid\; T!$$

$$\begin{array}{rcl}
[\![\epsilon]\!] & = & \{\epsilon\} \\
[\![a\,[m..n]]\!] & = & \{w \mid S(w) = \{a\},\; m \leq |w| \leq n\} \\
[\![T_1 + T_2]\!] & = & [\![T_1]\!] \cup [\![T_2]\!] \\
[\![T_1 \cdot T_2]\!] & = & [\![T_1]\!] \cdot [\![T_2]\!] \\
[\![T!]\!] & = & [\![T]\!] \setminus \{\epsilon\}
\end{array}$$

Conflict-Free REs, first restriction

$$T ::= \quad \epsilon \;\mid\; a\,[m..n] \;\mid\; T + T \;\mid\; T \cdot T \;\mid\; T \& T \;\mid\; T!$$

$$\begin{array}{rcl}
[\![\epsilon]\!] & = & \{\epsilon\} \\
[\![a\,[m..n]]\!] & = & \{w \mid S(w) = \{a\},\ m \leq |w| \leq n\} \\
[\![T_1 + T_2]\!] & = & [\![T_1]\!] \cup [\![T_2]\!] \\
[\![T_1 \cdot T_2]\!] & = & [\![T_1]\!] \cdot [\![T_2]\!] \\
[\![T!]\!] & = & [\![T]\!] \setminus \{\epsilon\} \\
[\![T_1 \& T_2]\!] & = & [\![T_1]\!] \& [\![T_2]\!]
\end{array}$$

Conflict-Free REs, first restriction

$$T ::= \epsilon \mid a[m..n] \mid T + T \mid T \cdot T \mid T \& T \mid T!$$

$$\begin{array}{rcl}
[\![\epsilon]\!] & = & \{\epsilon\} \\
[\![a[m..n]]\!] & = & \{w \mid S(w) = \{a\},\ m \leq |w| \leq n\} \\
[\![T_1 + T_2]\!] & = & [\![T_1]\!] \cup [\![T_2]\!] \\
[\![T_1 \cdot T_2]\!] & = & [\![T_1]\!] \cdot [\![T_2]\!] \\
[\![T!]\!] & = & [\![T]\!] \setminus \{\epsilon\} \\
[\![T_1 \& T_2]\!] & = & [\![T_1]\!] \& [\![T_2]\!]
\end{array}$$

Example :

$$\{ab\}\&\{XY\} = \{abXY, aXbY, aXYb, XabY, XaYb, XYab\}$$

permutations must respect order-constraints ! $XbYa \notin \{ab\}\&\{XY\}$

Conflict-Free REs, second restriction

[Conflict-free types] A type T is *conflict-free* if for any two distinct subterms $a\,[m..n]$ and $a'\,[m'..n']$ that occur in T, a is different from a'.

Conflict-Free REs, second restriction

[Conflict-free types] A type T is *conflict-free* if for any two distinct subterms $a\,[m..n]$ and $a'\,[m'..n']$ that occur in T, a is different from a'.

Examples:

- $(a\,[1..1]\,\&\; b\,[1..1]) + (a\,[1..1]\,\& c\,[1..1])$ is not CF

Conflict-Free REs, second restriction

[Conflict-free types] A type T is *conflict-free* if for any two distinct subterms $a\,[m..n]$ and $a'\,[m'..n']$ that occur in T, a is different from a'.

Examples:

- $(a\,[1..1]\,\&\; b\,[1..1]) + (a\,[1..1]\,\&c\,[1..1])$ is not CF
- The equivalent type $a\,[1..1]\,\&(b\,[1..1] + c\,[1..1])$ is CF.

Coding most used REs

- As shown in several recent papers, the *single occurrence* property is enjoyed by most XML schemas

Coding most used REs

- As shown in several recent papers, the *single occurrence* property is enjoyed by most XML schemas
- Repetition types T^* or T^+ are generalised by counting, which, is, however, restricted to symbols, so that, for example, types like $(a \cdot b)^*$ cannot be expressed.

Coding most used REs

- As shown in several recent papers, the *single occurrence* property is enjoyed by most XML schemas
- Repetition types T^* or T^+ are generalised by counting, which, is, however, restricted to symbols, so that, for example, types like $(a \cdot b)^*$ cannot be expressed.
- However, it has been found that DTD and XSD schemas use repetition almost exclusively as a^{op} or as $(a + \ldots + z)^{\mathrm{op}}$

Coding most used REs

- As shown in several recent papers, the *single occurrence* property is enjoyed by most XML schemas
- Repetition types T^* or T^+ are generalised by counting, which, is, however, restricted to symbols, so that, for example, types like $(a \cdot b)^*$ cannot be expressed.
- However, it has been found that DTD and XSD schemas use repetition almost exclusively as a^{op} or as $(a + \ldots + z)^{\mathrm{op}}$
- $(a + \ldots + z)^*$ can be expressed as $(a^* \& \ldots \& z^*)$, where a^* is a shortcut for $a\,[1..*] + \epsilon$

Coding most used REs

- As shown in several recent papers, the *single occurrence* property is enjoyed by most XML schemas
- Repetition types T^* or T^+ are generalised by counting, which, is, however, restricted to symbols, so that, for example, types like $(a \cdot b)^*$ cannot be expressed.
- However, it has been found that DTD and XSD schemas use repetition almost exclusively as a^{op} or as $(a + \ldots + z)^{\mathrm{op}}$
- $(a + \ldots + z)^*$ can be expressed as $(a^* \& \ldots \& z^*)$, where a^* is a shortcut for $a\,[1..*] + \epsilon$
- $(a + \ldots + z)^+$ can be expressed as $(a^* \& \ldots \& z^*)!$.

Coding most used REs

- As shown in several recent papers, the *single occurrence* property is enjoyed by most XML schemas
- Repetition types T^* or T^+ are generalised by counting, which, is, however, restricted to symbols, so that, for example, types like $(a \cdot b)^*$ cannot be expressed.
- However, it has been found that DTD and XSD schemas use repetition almost exclusively as a^{op} or as $(a + \ldots + z)^{\mathrm{op}}$
- $(a + \ldots + z)^*$ can be expressed as $(a^* \& \ldots \& z^*)$, where a^* is a shortcut for $a[1..*] + \epsilon$
- $(a + \ldots + z)^+$ can be expressed as $(a^* \& \ldots \& z^*)!$.

Coding most used REs

- As shown in several recent papers, the *single occurrence* property is enjoyed by most XML schemas
- Repetition types T^* or T^+ are generalised by counting, which, is, however, restricted to symbols, so that, for example, types like $(a \cdot b)^*$ cannot be expressed.
- However, it has been found that DTD and XSD schemas use repetition almost exclusively as a^{op} or as $(a + \ldots + z)^{\mathrm{op}}$
- $(a + \ldots + z)^*$ can be expressed as $(a^* \& \ldots \& z^*)$, where a^* is a shortcut for $a\,[1..*] + \epsilon$
- $(a + \ldots + z)^+$ can be expressed as $(a^* \& \ldots \& z^*)!$.

Hence, C-F types cover a wide class of REs used in human-designed XML schema.

Main ingredient : Types as constraints

We transform each CF type into an equivalent set of constraints.

Main ingredient : Types as constraints

We transform each CF type into an equivalent set of constraints.

For instance, consider $T = (a\,[1..3] \cdot b\,[2..2]) + c\,[1..2]$, and $w \in [\![T]\!]$:

Main ingredient : Types as constraints

We transform each CF type into an equivalent set of constraints.

For instance, consider $T = (a\,[1..3] \cdot b\,[2..2]) + c\,[1..2]$, and $w \in [\![T]\!]$:

1. **lower-bound**: at least one of a, b, and c appears in w;

Main ingredient : Types as constraints

We transform each CF type into an equivalent set of constraints.

For instance, consider $T = (a\,[1..3] \cdot b\,[2..2]) + c\,[1..2]$, and $w \in [\![T]\!]$:

1. **lower-bound**: at least one of a, b, and c appears in w;
2. **cardinality**: if a is in w, it appears 1, 2 or 3 times; if b is there, it appears twice; if c is there, it appears once or twice;

Main ingredient : Types as constraints

We transform each CF type into an equivalent set of constraints.

For instance, consider $T = (a\,[1..3] \cdot b\,[2..2]) + c\,[1..2]$, and $w \in [\![T]\!]$:

1. **lower-bound**: at least one of a, b, and c appears in w;
2. **cardinality**: if a is in w, it appears 1, 2 or 3 times; if b is there, it appears twice; if c is there, it appears once or twice;
3. **upper-bound**: no symbol out of $\{a, b, c\}$ is in w;

Main ingredient : Types as constraints

We transform each CF type into an equivalent set of constraints.

For instance, consider $T = (a\,[1..3] \cdot b\,[2..2]) + c\,[1..2]$, and $w \in [\![T]\!]$:

1. **lower-bound**: at least one of a, b, and c appears in w;
2. **cardinality**: if a is in w, it appears 1, 2 or 3 times; if b is there, it appears twice; if c is there, it appears once or twice;
3. **upper-bound**: no symbol out of $\{a, b, c\}$ is in w;
4. **exclusion**: if one of a, b is in w, then c is not, and if c is in w then neither of a, b is in w;

Main ingredient : Types as constraints

We transform each CF type into an equivalent set of constraints.

For instance, consider $T = (a\,[1..3] \cdot b\,[2..2]) + c\,[1..2]$, and $w \in [\![T]\!]$:

1. **lower-bound**: at least one of a, b, and c appears in w;
2. **cardinality**: if a is in w, it appears 1, 2 or 3 times; if b is there, it appears twice; if c is there, it appears once or twice;
3. **upper-bound**: no symbol out of $\{a, b, c\}$ is in w;
4. **exclusion**: if one of a, b is in w, then c is not, and if c is in w then neither of a, b is in w;
5. **co-occurrence**: if a is in w, then b is in w, and vice versa;

Main ingredient : Types as constraints

We transform each CF type into an equivalent set of constraints.

For instance, consider $T = (a\,[1..3] \cdot b\,[2..2]) + c\,[1..2]$, and $w \in [\![T]\!]$:

1. **lower-bound**: at least one of a, b, and c appears in w;
2. **cardinality**: if a is in w, it appears 1, 2 or 3 times; if b is there, it appears twice; if c is there, it appears once or twice;
3. **upper-bound**: no symbol out of $\{a, b, c\}$ is in w;
4. **exclusion**: if one of a, b is in w, then c is not, and if c is in w then neither of a, b is in w;
5. **co-occurrence**: if a is in w, then b is in w, and vice versa;
6. **order**: no occurrence of a may follow an occurrence of b.

The constraint language

Constraints are expressed using the following logic, where $a, b \in \Sigma$ and $A, B \subseteq \Sigma$:

$$\begin{array}{rcl} F & ::= & A^+ \ \mid \ A^+ \Mapsto B^+ \ \mid \ a?[m..n] \ \mid \ \mathrm{upper}(A) \\ & & \mid \ a \prec b \ \mid \ F \wedge F' \ \mid \ \mathbf{true} \end{array}$$

We do not have disjunction, nor negation.

Constraint semantics

$w \models A^{+} \quad \Leftrightarrow \quad S(w) \cap A \neq \emptyset$, i.e. some $a \in A$ appears in w

Constraint semantics

$w \models A^+$	$\Leftrightarrow$	$S(w) \cap A \neq \emptyset$, i.e. some $a \in A$ appears in w
$w \models A^+ \Mapsto B^+$	$\Leftrightarrow$	$w \not\models A^+$ or $w \models B^+$

Constraint semantics

$w \models A^+$	$\Leftrightarrow$	$S(w) \cap A \neq \emptyset$, i.e. some $a \in A$ appears in w
$w \models A^+ \Rrightarrow B^+$	$\Leftrightarrow$	$w \not\models A^+$ or $w \models B^+$
$w \models a?[m..n]$ $(n \neq *)$	$\Leftrightarrow$	if a appears in w, then it appears at least m times and at most n times

Constraint semantics

$w \models A^+$	$\Leftrightarrow$	$S(w) \cap A \neq \emptyset$, i.e. some $a \in A$ appears in w
$w \models A^+ \Rrightarrow B^+$	$\Leftrightarrow$	$w \not\models A^+$ or $w \models B^+$
$w \models a?[m..n]$ $(n \neq *)$	$\Leftrightarrow$	if a appears in w, then it appears at least m times and at most n times
$w \models a?[m..*]$	$\Leftrightarrow$	if a appears in w, then it appears at least m times

Constraint semantics

$w \models A^+$	$\Leftrightarrow$	$S(w) \cap A \neq \emptyset$, i.e. some $a \in A$ appears in w
$w \models A^+ \Rrightarrow B^+$	$\Leftrightarrow$	$w \not\models A^+$ or $w \models B^+$
$w \models a?[m..n]$ $(n \neq *)$	$\Leftrightarrow$	if a appears in w, then it appears at least m times and at most n times
$w \models a?[m..*]$	$\Leftrightarrow$	if a appears in w, then it appears at least m times
$w \models \mathrm{upper}(A)$	$\Leftrightarrow$	$S(w) \subseteq A$

Constraint semantics

$w \models A^+$	$\Leftrightarrow$	$S(w) \cap A \neq \emptyset$, i.e. some $a \in A$ appears in w
$w \models A^+ \Mapsto B^+$	$\Leftrightarrow$	$w \not\models A^+$ or $w \models B^+$
$w \models a?[m..n]\ (n \neq *)$	$\Leftrightarrow$	if a appears in w, then it appears at least m times and at most n times
$w \models a?[m..*]$	$\Leftrightarrow$	if a appears in w, then it appears at least m times
$w \models \mathrm{upper}(A)$	$\Leftrightarrow$	$S(w) \subseteq A$
$w \models a \prec b$	$\Leftrightarrow$	there is no occurrence of a in w that follows an occurrence of b in w

Constraint semantics

$w \models A^+$	$\Leftrightarrow$	$S(w) \cap A \neq \emptyset$, i.e. some $a \in A$ appears in w
$w \models A^+ \Rrightarrow B^+$	$\Leftrightarrow$	$w \not\models A^+$ or $w \models B^+$
$w \models a?[m..n]$ $(n \neq *)$	$\Leftrightarrow$	if a appears in w, then it appears at least m times and at most n times
$w \models a?[m..*]$	$\Leftrightarrow$	if a appears in w, then it appears at least m times
$w \models \mathrm{upper}(A)$	$\Leftrightarrow$	$S(w) \subseteq A$
$w \models a \prec b$	$\Leftrightarrow$	there is no occurrence of a in w that follows an occurrence of b in w
$w \models F_1 \wedge F_2$	$\Leftrightarrow$	$w \models F_1$ and $w \models F_2$

Constraint semantics

$w \models A^+$	$\Leftrightarrow$	$S(w) \cap A \neq \emptyset$, i.e. some $a \in A$ appears in w
$w \models A^+ \Mapsto B^+$	$\Leftrightarrow$	$w \not\models A^+$ or $w \models B^+$
$w \models a?[m..n]\ (n \neq *)$	$\Leftrightarrow$	if a appears in w, then it appears at least m times and at most n times
$w \models a?[m..*]$	$\Leftrightarrow$	if a appears in w, then it appears at least m times
$w \models \mathrm{upper}(A)$	$\Leftrightarrow$	$S(w) \subseteq A$
$w \models a \prec b$	$\Leftrightarrow$	there is no occurrence of a in w that follows an occurrence of b in w
$w \models F_1 \wedge F_2$	$\Leftrightarrow$	$w \models F_1$ and $w \models F_2$
$w \models \mathbf{true}$	$\Leftrightarrow$	always

examples

$$\epsilon \not\models A^+ \qquad \epsilon \models \mathrm{upper}(A) \qquad \epsilon \models a?[m..n]$$

$$\epsilon \models a \prec b \qquad b \models a \prec b \qquad aba \not\models a \prec b$$

$$w \not\models \emptyset^+ \qquad w \models \emptyset^+ \Rrightarrow A^+ \qquad w \models \emptyset^+ \Rrightarrow \emptyset^+$$

some abbreviations

$$
\begin{aligned}
a^{+} \quad &=_{def} \quad \{a\}^{+} \\
a \prec\succ b \quad &=_{def} \quad (a \prec b) \wedge (b \prec a) \\
A \prec B \quad &=_{def} \quad \bigwedge_{a \in A, b \in B} a \prec b \\
A \prec\succ B \quad &=_{def} \quad \bigwedge_{a \in A, b \in B} a \prec\succ b
\end{aligned}
$$

some abbreviations

$$
\begin{aligned}
a^+ &=_{def} \{a\}^+ \\
a \prec\succ b &=_{def} (a \prec b) \wedge (b \prec a) \\
A \prec B &=_{def} \bigwedge_{a \in A, b \in B} a \prec b \\
A \prec\succ B &=_{def} \bigwedge_{a \in A, b \in B} a \prec\succ b
\end{aligned}
$$

Exclusion is expressed in terms of order-constraints!

some abbreviations

$$
\begin{array}{rcl}
a^{+} & =_{def} & \{a\}^{+} \\
a \prec\succ b & =_{def} & (a \prec b) \wedge (b \prec a) \\
A \prec B & =_{def} & \bigwedge_{a \in A, b \in B} a \prec b \\
A \prec\succ B & =_{def} & \bigwedge_{a \in A, b \in B} a \prec\succ b
\end{array}
$$

Exclusion is expressed in terms of order-constraints!

Also

$$T \models F \;\Leftrightarrow\; \forall w \in [\![T]\!].\ w \models F$$

Main Theorem

For each CF type T:

$$w \in [\![T]\!] \quad \Leftrightarrow \quad w \models \mathcal{FC}(T) \wedge \mathcal{NC}(T)$$

$\mathcal{FC}(T)$ is the conjunction of flat constraints associated to T

$\mathcal{NC}(T)$ is the conjunction of nested constraints associated to T

In $\mathcal{FC}(T)$ and $\mathcal{NC}(T)$ construction, a central role is played by the property $\mathrm{N}(T)$, which holds iff $\epsilon \in [\![T]\!]$

$\mathrm{N}(T)$ can be checked in linear time

Flat constraints $\mathcal{FC}(T)$

Lower-bound:	$SIf(T)$	$=_{def}$	$If_T(S^+(T))$
Cardinality:	$\mathrm{ZeroMinMax}(T)$	$=_{def}$	$\bigwedge_{a[m..n]\in Atoms(T)} a?[m..n]$
Upper-bound:	$\mathrm{upperS}(T)$	$=_{def}$	$\mathrm{upper}(S(T))$
Flat constraints:	$\mathcal{FC}(T)$	$=_{def}$	$SIf(T) \wedge \mathrm{ZeroMinMax}(T) \wedge$

Nested Constraints $\mathcal{NC}(T)$

Co-occurrence:

$$\mathcal{CC}(T_1 + T_2) \quad =_{def} \quad \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2)$$

Nested Constraints $\mathcal{NC}(T)$

Co-occurrence:

$$\mathcal{CC}(T_1 + T_2) =_{def} \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2)$$
$$\mathcal{CC}(T_1 \otimes T_2) =_{def} If_{T_2}(S^+(T_1) \Mapsto S^+(T_2))$$

Nested Constraints $\mathcal{NC}(T)$

Co-occurrence:

$$
\begin{aligned}
\mathcal{CC}(T_1 + T_2) &=_{def} \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2) \\
\mathcal{CC}(T_1 \otimes T_2) &=_{def} \mathit{If}_{T_2}(S^+(T_1) \Rrightarrow S^+(T_2)) \\
&\quad \wedge \mathit{If}_{T_1}(S^+(T_2) \Rrightarrow S^+(T_1)) \\
&\quad \wedge \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2)
\end{aligned}
$$

Nested Constraints $\mathcal{NC}(T)$

Co-occurrence:

$$
\begin{aligned}
\mathcal{CC}(T_1 + T_2) &=_{def} \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2) \\
\mathcal{CC}(T_1 \otimes T_2) &=_{def} \mathit{If}_{T_2}(S^+(T_1) \Mapsto S^+(T_2)) \\
&\quad \wedge \mathit{If}_{T_1}(S^+(T_2) \Mapsto S^+(T_1)) \\
&\quad \wedge \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2) \\
\mathcal{CC}(T!) &=_{def} \mathcal{CC}(T)
\end{aligned}
$$

Nested Constraints $\mathcal{NC}(T)$

Co-occurrence:

$$
\begin{array}{lll}
\mathcal{CC}(T_1 + T_2) & =_{def} & \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2) \\
\mathcal{CC}(T_1 \otimes T_2) & =_{def} & \mathit{If}_{T_2}(S^+(T_1) \Rrightarrow S^+(T_2)) \\
 & & \wedge\ \mathit{If}_{T_1}(S^+(T_2) \Rrightarrow S^+(T_1)) \\
 & & \wedge\ \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2) \\
\mathcal{CC}(T!) & =_{def} & \mathcal{CC}(T) \\
\mathcal{CC}(\epsilon) =_{def} \mathcal{CC}(a\,[m..n]) & =_{def} & \textbf{true}
\end{array}
$$

Nested Constraints $\mathcal{NC}(T)$

Co-occurrence:

$$
\begin{array}{lcl}
\mathcal{CC}(T_1 + T_2) & =_{def} & \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2) \\
\mathcal{CC}(T_1 \otimes T_2) & =_{def} & \mathit{If}_{T_2}(S^+(T_1) \Mapsto S^+(T_2)) \\
 & & \wedge\ \mathit{If}_{T_1}(S^+(T_2) \Mapsto S^+(T_1)) \\
 & & \wedge\ \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2) \\
\mathcal{CC}(T!) & =_{def} & \mathcal{CC}(T) \\
\mathcal{CC}(\epsilon) =_{def} \mathcal{CC}(a\,[m..n]) & =_{def} & \mathbf{true}
\end{array}
$$

Order and exclusion:

$$
\begin{array}{lcl}
\mathcal{OC}(T_1 + T_2) & =_{def} & (S(T_1) \prec\succ S(T_2)) \wedge \mathcal{OC}(T_1) \wedge \mathcal{OC}(T_2)
\end{array}
$$

Nested Constraints $\mathcal{NC}(T)$

Co-occurrence:

$$
\begin{array}{lll}
\mathcal{CC}(T_1 + T_2) & =_{def} & \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2) \\
\mathcal{CC}(T_1 \otimes T_2) & =_{def} & \mathit{If}_{T_2}(S^+(T_1) \Mapsto S^+(T_2)) \\
 & & \wedge\ \mathit{If}_{T_1}(S^+(T_2) \Mapsto S^+(T_1)) \\
 & & \wedge\ \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2) \\
\mathcal{CC}(T!) & =_{def} & \mathcal{CC}(T) \\
\mathcal{CC}(\epsilon) =_{def} \mathcal{CC}(a\,[m..n]) & =_{def} & \mathbf{true}
\end{array}
$$

Order and exclusion:

$$
\begin{array}{lll}
\mathcal{OC}(T_1 + T_2) & =_{def} & (S(T_1) \prec\succ S(T_2)) \wedge \mathcal{OC}(T_1) \wedge \mathcal{OC}(T_2) \\
\mathcal{OC}(T_1 \& T_2) & =_{def} & \mathcal{OC}(T_1) \wedge \mathcal{OC}(T_2)
\end{array}
$$

Nested Constraints $\mathcal{NC}(T)$

Co-occurrence:

$$
\begin{array}{lll}
\mathcal{CC}(T_1 + T_2) & =_{def} & \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2) \\
\mathcal{CC}(T_1 \otimes T_2) & =_{def} & \mathit{If}_{T_2}(S^+(T_1) \Mapsto S^+(T_2)) \\
 & & \wedge\ \mathit{If}_{T_1}(S^+(T_2) \Mapsto S^+(T_1)) \\
 & & \wedge\ \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2) \\
\mathcal{CC}(T!) & =_{def} & \mathcal{CC}(T) \\
\mathcal{CC}(\epsilon) =_{def} \mathcal{CC}(a\,[m..n]) & =_{def} & \mathbf{true}
\end{array}
$$

Order and exclusion:

$$
\begin{array}{lll}
\mathcal{OC}(T_1 + T_2) & =_{def} & (S(T_1) \prec\succ S(T_2)) \wedge \mathcal{OC}(T_1) \wedge \mathcal{OC}(T_2) \\
\mathcal{OC}(T_1 \& T_2) & =_{def} & \mathcal{OC}(T_1) \wedge \mathcal{OC}(T_2) \\
\mathcal{OC}(T_1 \cdot T_2) & =_{def} & (S(T_1) \prec S(T_2)) \wedge \mathcal{OC}(T_1) \wedge \mathcal{OC}(T_2)
\end{array}
$$

Nested Constraints $\mathcal{NC}(T)$

Co-occurrence:

$$
\begin{array}{lll}
\mathcal{CC}(T_1 + T_2) & =_{def} & \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2) \\
\mathcal{CC}(T_1 \otimes T_2) & =_{def} & \mathit{If}_{T_2}(S^+(T_1) \Mapsto S^+(T_2)) \\
 & & \wedge\ \mathit{If}_{T_1}(S^+(T_2) \Mapsto S^+(T_1)) \\
 & & \wedge\ \mathcal{CC}(T_1) \wedge \mathcal{CC}(T_2) \\
\mathcal{CC}(T!) & =_{def} & \mathcal{CC}(T) \\
\mathcal{CC}(\epsilon) =_{def} \mathcal{CC}(a\,[m..n]) & =_{def} & \mathbf{true}
\end{array}
$$

Order and exclusion:

$$
\begin{array}{lll}
\mathcal{OC}(T_1 + T_2) & =_{def} & (S(T_1) \prec\succ S(T_2)) \wedge \mathcal{OC}(T_1) \wedge \mathcal{OC}(T_2) \\
\mathcal{OC}(T_1 \& T_2) & =_{def} & \mathcal{OC}(T_1) \wedge \mathcal{OC}(T_2) \\
\mathcal{OC}(T_1 \cdot T_2) & =_{def} & (S(T_1) \prec S(T_2)) \wedge \mathcal{OC}(T_1) \wedge \mathcal{OC}(T_2) \\
\mathcal{OC}(T!) & =_{def} & \mathcal{OC}(T) \\
\mathcal{OC}(\epsilon) =_{def} \mathcal{OC}(a\,[m..n]) & =_{def} & \mathbf{true}
\end{array}
$$

Nested constraints:

$$
\mathcal{NC}(T) =_{def} \mathcal{CC}(T) \wedge \mathcal{OC}(T)
$$

Subtyping as constraints satisfation

From

$$w \in [\![T]\!] \quad \Leftrightarrow \quad w \models \mathcal{FC}(T) \wedge \mathcal{CC}(T) \wedge \mathcal{OC}(T)$$

Subtyping as constraints satisfation

From

$$w \in [\![T]\!] \quad \Leftrightarrow \quad w \models \mathcal{FC}(T) \wedge \mathcal{CC}(T) \wedge \mathcal{OC}(T)$$

the following property directly follows:

$$[\![T]\!] \subseteq [\![U]\!] \quad \Leftrightarrow \quad T \models \mathcal{CC}(U) \quad \wedge \quad T \models \mathcal{OC}(U) \quad \wedge T \models \mathcal{FC}(U)$$

Subtyping as constraints satisfation

From

$$w \in [\![T]\!] \quad \Leftrightarrow \quad w \models \mathcal{FC}(T) \wedge \mathcal{CC}(T) \wedge \mathcal{OC}(T)$$

the following property directly follows:

$$[\![T]\!] \subseteq [\![U]\!] \quad \Leftrightarrow \quad T \models \mathcal{CC}(U) \quad \wedge \quad T \models \mathcal{OC}(U) \quad \wedge T \models \mathcal{FC}(U)$$

We provide a polynomial type inclusion algorithm, by providing three polynomial algorithms for checking $T \models \mathcal{CC}(U)$, $T \models \mathcal{OC}(U)$ and $T \models \mathcal{FC}(U)$, respectively.

Subtyping as constraints satisfation

From

$$w \in [\![T]\!] \quad \Leftrightarrow \quad w \models \mathcal{FC}(T) \wedge \mathcal{CC}(T) \wedge \mathcal{OC}(T)$$

the following property directly follows:

$$[\![T]\!] \subseteq [\![U]\!] \quad \Leftrightarrow \quad T \models \mathcal{CC}(U) \quad \wedge \quad T \models \mathcal{OC}(U) \quad \wedge T \models \mathcal{FC}(U)$$

We provide a polynomial type inclusion algorithm, by providing three polynomial algorithms for checking $T \models \mathcal{CC}(U)$, $T \models \mathcal{OC}(U)$ and $T \models \mathcal{FC}(U)$, respectively.

We will focus on $T \models \mathcal{CC}(U)$ (see the paper for details on the other two algorithms)

Polynomial checking of $T \models \mathcal{CC}(U)$

$\mathcal{CC}(U)$ is a finite conjunction of at most $2 \times |U|$ constraints of the form $A^+ \Rrightarrow B^+$

Polynomial checking of $T \models \mathcal{CC}(U)$

$\mathcal{CC}(U)$ is a finite conjunction of at most $2 \times |U|$ constraints of the form $A^+ \Mapsto B^+$

Given $A^+ \Mapsto B^+ \in \mathcal{CC}(U)$, in order to check $T \models A^+ \Mapsto B^+$ we first compute the 'backward closure"

$$\mathrm{BC}(B)_T = \{a \in S(T) \mid T \models a^+ \Mapsto B^+\}$$

Polynomial checking of $T \models \mathcal{CC}(U)$

$\mathcal{CC}(U)$ is a finite conjunction of at most $2 \times |U|$ constraints of the form $A^+ \mapsto B^+$

Given $A^+ \mapsto B^+ \in \mathcal{CC}(U)$, in order to check $T \models A^+ \mapsto B^+$ we first compute the ‘backward closure”

$$\mathrm{BC}(B)_T = \{a \in S(T) \mid T \models a^+ \mapsto B^+\}$$

and then we are done since it turns out that

$$\begin{aligned} & T \models A^+ \mapsto B^+ \\ \Leftrightarrow\ & \forall a \in (A \cap S(T)).\ T \models a^+ \mapsto B^+ \\ \Leftrightarrow\ & (A \cap S(T)) \subseteq \mathrm{BC}(B)_T \end{aligned}$$

Computing $\mathrm{BC}(B)_T$

We compute $B_T^{\uparrow}$ such that T' is a subterm of T and $T' \in B_T^{\uparrow}$ entails $T \models S(T')^+ \mapsto B^+$

Computing $\mathrm{BC}(B)_T$

We compute $B^{\uparrow}_T$ such that T' is a subterm of T and $T' \in B^{\uparrow}_T$ entails $T \models S(T')^+ \mapsto B^+$

And then we show that $\mathrm{BC}(B)_T = \cup_{T' \in B^{\uparrow}_T} S(T')$.

Computing $\mathrm{BC}(B)_T$

We compute $B^{\uparrow}_T$ such that T' is a subterm of T and $T' \in B^{\uparrow}_T$ entails $T \models S(T')^+ \Rrightarrow B^+$

And then we show that $\mathrm{BC}(B)_T = \cup_{T' \in B^{\uparrow}_T} S(T')$.

$$
\begin{array}{rlcl}
(\epsilon) & \text{for any } B, T & & \epsilon \in B^{\uparrow}_T \\
(a\,[m..n]) & T = C[a\,[m..n]],\ a \in B & \Rightarrow & a\,[m..n] \in B^{\uparrow}_T \\
(+) & T = C[T_1 + T_2],\ T_1 \in B^{\uparrow}_T,\ T_2 \in B^{\uparrow}_T & \Rightarrow & T_1 + T_2 \in B^{\uparrow}_T \\
(\otimes) & T = C[T_1 \otimes T_2],\ T_1 \in B^{\uparrow}_T,\ T_2 \in B^{\uparrow}_T & \Rightarrow & T_1 \otimes T_2 \in B^{\uparrow}_T \\
(\otimes \Rrightarrow) & T = C[T_1 \otimes T_2],\ \neg\mathrm{N}(T_2),\ T_2 \in B^{\uparrow}_T & \Rightarrow & T_1 \otimes T_2 \in B^{\uparrow}_T \\
(\otimes \Lleftarrow) & T = C[T_1 \otimes T_2],\ \neg\mathrm{N}(T_1),\ T_1 \in B^{\uparrow}_T & \Rightarrow & T_1 \otimes T_2 \in B^{\uparrow}_T \\
(!) & T = C[T_1!],\ T_1 \in B^{\uparrow}_T & \Rightarrow & T_1! \in B^{\uparrow}_T
\end{array}
$$

$\mathrm{BC}(B)_T$ can be computed in $O(|B| + |T|)$, hence $O(|U| + |T|)$, time.

Complexity of $T \models \mathcal{CC}(U)$

We have that :

- $\mathcal{CC}(U)$ has at most $2 \times |U|$ constraints of the form $A^+ \Mapsto B^+$.

Complexity of $T \models \mathcal{CC}(U)$

We have that :

- $\mathcal{CC}(U)$ has at most $2 \times |U|$ constraints of the form $A^+ \Mapsto B^+$.
- $T \models A^+ \Mapsto B^+ \Leftrightarrow (A \cap S(T)) \subseteq \ \mathrm{BC}(B)_T$

Complexity of $T \models \mathcal{CC}(U)$

We have that :

- $\mathcal{CC}(U)$ has at most $2 \times |U|$ constraints of the form $A^+ \Mapsto B^+$.
- $T \models A^+ \Mapsto B^+ \Leftrightarrow (A \cap S(T)) \subseteq \mathrm{BC}(B)_T$
- We can check $(A \cap S(T)) \subseteq \mathrm{BC}(B)_T$ in $O(|U| + |T|)$ time.

Complexity of $T \models \mathcal{CC}(U)$

We have that :

- $\mathcal{CC}(U)$ has at most $2 \times |U|$ constraints of the form $A^+ \Mapsto B^+$.
- $T \models A^+ \Mapsto B^+ \Leftrightarrow (A \cap S(T)) \subseteq \mathrm{BC}(B)_T$
- We can check $(A \cap S(T)) \subseteq \mathrm{BC}(B)_T$ in $O(|U| + |T|)$ time.

So $T \models \mathcal{CC}(U)$ can be checked in $O(|U| \times |T| + |U|^2)$ time.

Since $T \models \mathcal{OC}(U)$ and $T \models \mathcal{FC}(U)$ have lower complexity, we have that $T < U$ can be checked in $O(|U| \times |T| + |U|^2)$ time.

Conclusion and Future Work

- In the paper, to appear in IS, we also prove that intersection on CF types is NP complete.
- In a recent ICDT paper we have provided a polynomial algorithm to check $T < U$ where only U is required to be CF
- We are looking for extensions with intersection admitting polynomial complexity
- We plan to define an hybrid algorithm, that uses the PTIME algorithm whenever possible, and resorts to the full algorithm when needed.