



CPF: The Certification Problem Format

René Thiemann University of Innsbruck

COST action CA20111, WG3 Kick-off meeting, February 10, 2022





tool

- running several tools
 - standard input format beneficial (DIMACS, TPTP, TPDB, ...)

Certification

universitat

innsbruck



- running several tools
 - standard input format beneficial (DIMACS, TPTP, TPDB, ...)
- certification: verify output of untrusted tools
 - use trusted certifier
 - requires certificates

Certification



- running several tools
 - standard input format beneficial (DIMACS, TPTP, TPDB, ...)
- certification: verify output of untrusted tools
 - use trusted certifier
 - requires certificates
- restriction of certificate format in this talk
 - fixed set of languages (term rewrite systems, integer transition systems)
 - fixed set of properties (termination, complexity, safety, ...)
 - fixed set of supported techniques

Situation before CPF: Multiple Formats for Certificates



Using the Certification Problem Format



Using the Certification Problem Format



Advantages of Certification Problem Format

- tools only have to support up to 2 output formats
- \Rightarrow certifiers are more likely to get input
 - new certifiers can be based on CPF
- \Rightarrow no need to convince tool authors to support just another output format
 - CPF can be used as "input problem" for certifiers
- \Rightarrow gather database of CPFs

• June 2009: initial version, termination proofs

joint development by teams of certifiers for term rewrite systems (Blanqui, Contejean, Sternagel, Thiemann, Urbain, ...)

- June 2009: initial version, termination proofs joint development by teams of certifiers for term rewrite systems (Blanqui, Contejean, Sternagel, Thiemann, Urbain, ...)
- since 2009: used for termination competition

- June 2009: initial version, termination proofs joint development by teams of certifiers for term rewrite systems (Blanqui, Contejean, Sternagel, Thiemann, Urbain, ...)
- since 2009: used for termination competition
- October 2011: confluence and completion proofs

- June 2009: initial version, termination proofs joint development by teams of certifiers for term rewrite systems (Blanqui, Contejean, Sternagel, Thiemann, Urbain, ...)
- since 2009: used for termination competition
- October 2011: confluence and completion proofs
- April 2012: complexity proofs

- June 2009: initial version, termination proofs joint development by teams of certifiers for term rewrite systems (Blanqui, Contejean, Sternagel, Thiemann, Urbain, ...)
- since 2009: used for termination competition
- October 2011: confluence and completion proofs
- April 2012: complexity proofs
- since 2012: used in confluence competition

- June 2009: initial version, termination proofs joint development by teams of certifiers for term rewrite systems (Blanqui, Contejean, Sternagel, Thiemann, Urbain, ...)
- since 2009: used for termination competition
- October 2011: confluence and completion proofs
- April 2012: complexity proofs
- since 2012: used in confluence competition
- since then, addition of several new input formats and properties
 - variants of rewriting, several properties
 - integer transition systems (termination and safety)
 - on its way: simplified LLVM (termination)

Design Choice: XML

- CPF generation supported by various tools
 - termination (AProVE, ConCon, CiME3, Matchbox, NaTT, TTT2, Microsoft T2)
 - confluence (ACP, CSI)
 - complexity (AProVE, CaT, TCT)
 - completion (mkbTT, KBCV)
 - safety (Microsoft T2)
- characteristics

universität

- small number of proofs steps
- single proof step may be complex
- new proof techniques frequently arise

(in comparison to SAT/SMT)

Design Choice: XML

- CPF generation supported by various tools
 - termination (AProVE, ConCon, CiME3, Matchbox, NaTT, TTT2, Microsoft T2)
 - confluence (ACP, CSI)
 - complexity (AProVE, CaT, TCT)
 - completion (mkbTT, KBCV)
 - safety (Microsoft T2)
- characteristics
 - small number of proofs steps
 - single proof step may be complex
 - new proof techniques frequently arise
- consequences: use XML
 - XML overhead does not harm
 - easy to produce and display via XML-libraries
 - easy to extend CPF by new definitions

(in comparison to SAT/SMT)

Example Complexity Proof: Source and Pretty-Printed

← → C D view-source:file:///Users/rene/Conferences/2022/COST_Meeting/demos/cate; ☆

C file:///Users/rene/Conferences/2022/COST_Meeting/demos/categories/comple ☆ ○

ccartificationProblem xmlms;xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:noNamespaceSchemaLocation="cpf.xsd"> <input> <complexityInput> «trsInput» stras <rules> <rule> <lhs> <funapp> sname>fs/name> <arg> stupport enaments (names <arg> evarave luara </are> </funance </arg> «/funapp> «/lhs> scrhs> <funances sname>fs/name> <arg> <funapp> ennamone/names CALIFY stunner. snames f </names sange syarsxs/yars </arg> </funapp> «/arm» «/funann» «/arm» «/funann» elebsa e/rules c/ruless eltras <strategy><innermost/></strategy> </trsingut> -derivationalComplexity> <signature> <symbol> <name>f</name> carity>1</arity> </symbol> <synbol> <name>g</name> saritys1s/aritys e/symbol's e/signatures c/derivationalComplexitys <polynomial>2</polynomial> </complexityInput>

Complexity Proof

by ttt2

0

Input

Derivational complexity of the following relation is considered. The intended complexity is $O(n^2)$. The following symbols are considered: f, g.

The rewrite relation of the following TRS is considered.

 $f(f(x)) \rightarrow f(g(f(x)))$ The evaluation strategy is innermost.

Proof

1 Rule Shifting

The rules (f(x)) - f(g(f(x))) g(f(x)) - f(g(f(x))) are strictly oriented by the following linear polynomial interpretation over (2 x 2)-matrices with strict dimension 1 over the rationals with due tas = 1/64

[g(x ₁)] -	1 0	0 0	· x1 +	4 0	0
[f(x ₁)] -	1	5 0] · x1 +	0 4	0 0

which has the intended complexity.

1.1 R is empty

There are no rules in the TRS R. Hence, R/S has complexity O(1).

Tool configuration

:::2

version: 1.08

· strategy: matrix -dim 2 -triangle -ib 3

Design Choice: Certifier-Friendly

- CPF created by representatives of certifiers by merging preliminary formats of certifiers
- \Rightarrow CPF is certifier-friendly
 - complete proof tree has to be given
 - some certifier needs information ⇒ CPF demands it (prefer verbose certificates over complicated reconstruction)
 - still CPF is easy to generate

```
<redPair> ... </redPair>
<usableRules> ... </usableRules> <!-- convenience -->
<dpProof> ... </dpProof>
</redPairUrProc>
</dpProof>
```

Design Choice: Determinism

- several variants of same technique
 - naming scheme of variables, new symbols, ...
 - approximation of well-founded orders
 - . . .

Design Choice: Determinism

- several variants of same technique
 - naming scheme of variables, new symbols, ...
 - approximation of well-founded orders

• ...

- specifying each possible variant is cumbersome (would require formalization of all variants)
- solution: leave variants unspecified, provide result in output

Example: Reduction Pair Processor

Theorem (Rule Removal)

innsbruck

if (\succeq,\succ) is reduction pair and $\mathcal{P} \cup \mathcal{U}(\mathcal{P},\mathcal{R}) \subseteq \succeq$ then $SN(\mathcal{P} \setminus \succ,\mathcal{R}) \implies SN(\mathcal{P},\mathcal{R})$

- problem: \succ usually only approximated
- $\Rightarrow \mathcal{P} \setminus \succ$ depends on approximation

```
<dpProof>
 <redPairUrProc>
   <redPair> ... </redPair>
   <usableRules> ... </usableRules>
   <dps> ... </dps> <!-- remaining pairs -->
   <dpProof> ... </dpProof>
 </redPairUrProc>
</dpProof>
```

Significance of Determinism

Key issue: identify wrong proof steps at right position

• wrong termination proof (in first step it was not allowed to delete pair 2)

$$(\{1,2,3\},\mathcal{R}) \stackrel{redpair}{\hookrightarrow} (\{3\},\mathcal{R}) \stackrel{depgraph}{\hookrightarrow} done$$

certifying proof without determinism

$$(\{1,2,3\},\mathcal{R}) \stackrel{redpair}{\hookrightarrow} (\{2,3\},\mathcal{R}) \stackrel{depgraph}{\hookrightarrow} not \ done$$

certifying proof with determinism

$$(\{1,2,3\},\mathcal{R}) \stackrel{redpair,\{3\}}{\hookrightarrow}$$
 not allowed to switch to $(\{3\},\mathcal{R})$

Extensions and Changes of CPF

- extensions
 - CPF was extended regularly (new definitions / techniques / properties)
 - since 2016: discussion in community via CPF mailing list
 - from 2016: discussion between CellA-team and tool authors
 - problem: some certifiers that supported CPF initially are no longer developed
- changes
 - are tried to be avoided (backward compatibility)
 - sometimes beneficial (fix design bugs, unify elements, ...)
 - so far only happened twice (CPF 1.x to 2.0, 2.0 to 2.1), converters available

Conclusion

- CPF
 - proof format with fixed set of properties and techniques
 - used in two annual competitions (currently checked by CeTA certifier)
 - supported by several tools
- benefits in two directions
 - certifiers: get inputs from various tools
 - tools: generated proofs get checked on validity
- availability of CPF
 - http://cl-informatik.uibk.ac.at/software/cpf/ (old: dedicated website)
 - http://cl-informatik.uibk.ac.at/software/ceta/
- (new: part of CeTA)

- problems
 - only one actively maintained certifier (?)
 - limited expressivity, not generic \Rightarrow interesting for WG3?