Congruence Closure of Compressed Terms in Polynomial Time

Manfred Schmidt-Schauß, David Sabel, Altug Anis

Goethe-University, Frankfurt am Main, Germany

FroCoS'11, Saarbrücken

Motivation

- Consider **compressed** representation of **terms** to optimize space (and time) usage
- Applications e.g. XML-trees and XML-processing
- Design efficient algorithms on compressed terms **without** prior **decompression**
- We use **tree grammars** as a clean representation of compressed terms

Some Previous / Related Work

- Polynomial **equality check** of grammar compressed **strings**:
  Plandowski '94, **Lifshits '07**
- **Equality check** of grammar compressed **terms**:
  Busatto, Lohrey, Maneth '05; **Schmidt-Schauß '05**
- **Compression** of XML documents using tree grammars:
  Busatto, Lohrey, Maneth '05
- **Unification** for grammar compressed **terms**:
  Gasćon, Godoy, & Schmidt-Schauß '08
- Analysis of **pattern matching** on compressed **terms**:
  Schmidt-Schauß '11

Our Contribution

- Combining **equational reasoning** with **grammar compression** for terms
- We consider the special case of **ground equations**
- In the **uncompressed** case efficiently decidable $O(n \log n)$ by **congruence closure** algorithms

Applications e.g. SMT solvers can e.g. deal with equational theories defined by a set of ground equations.

Compressed Representation of Terms

Singleton tree grammars (STG): $G = (\mathcal{TN}, \mathcal{CN}, \Sigma, R)$

- $\mathcal{TN}$ term nonterminals
- Σ signature of function symbols
- $\mathcal{CN}$ context nonterminals
- R production rules

Compressed Representation of Terms

Singleton tree grammars (STG): $G = (\mathcal{TN}, \mathcal{CN}, \Sigma, R)$

- $\mathcal{TN}$ term nonterminals
- Σ signature of function symbols
- $\mathcal{CN}$ context nonterminals
- R production rules

Side-conditions

- for every $A \in \mathcal{TN}$: $\mathrm{val}_G(A) \in \mathcal{T}(\Sigma)$;
  for every $C \in \mathcal{C}$: $\mathrm{val}_G(C)$ is a context on $\mathcal{T}(\Sigma)$
- R is acyclic and has exactly one rule for every nonterminal
- Allowed rules in R ($A, A_i \in \mathcal{TN}$; $C, C_i \in \mathcal{CN}$; $f \in \Sigma$)

$$\boldsymbol{A} ::= \boldsymbol{f}(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m) \qquad \boldsymbol{A}_1 ::= \boldsymbol{A}_2 \qquad \boldsymbol{A}_1 ::= \boldsymbol{C}_1[\boldsymbol{A}_2]$$

$$\boldsymbol{C} ::= \boldsymbol{f}(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_i, [\cdot], \boldsymbol{A}_{i+2}, \ldots, \boldsymbol{A}_m) \qquad \boldsymbol{C} ::= [\cdot] \qquad \boldsymbol{C} ::= \boldsymbol{C}_1[\boldsymbol{C}_2]$$

Compressed Representation of Terms

Singleton tree grammars (STG): $G = (\mathcal{TN}, \mathcal{CN}, \Sigma, R)$

- $\mathcal{TN}$ term nonterminals
- Σ signature of function symbols
- $\mathcal{CN}$ context nonterminals
- R production rules

Side-conditions

- for every $A \in \mathcal{TN}$: $\mathrm{val}_G(A) \in \mathcal{T}(\Sigma)$;
  for every $C \in \mathcal{C}$: $\mathrm{val}_G(C)$ is a context on $\mathcal{T}(\Sigma)$
- R is acyclic and has exactly one rule for every nonterminal
- Allowed rules in R ($A, A_i \in \mathcal{TN}$; $C, C_i \in \mathcal{CN}$; $f \in \Sigma$)

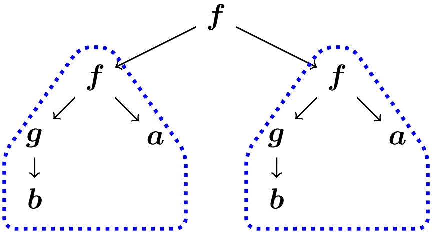
$$\boldsymbol{A} ::= \boldsymbol{f}(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_m) \qquad \boldsymbol{A}_1 ::= \boldsymbol{A}_2 \qquad \boldsymbol{A}_1 ::= \boldsymbol{C}_1[\boldsymbol{A}_2]$$

$$\boldsymbol{C} ::= \boldsymbol{f}(\boldsymbol{A}_1, \ldots, \boldsymbol{A}_i, [\cdot], \boldsymbol{A}_{i+2}, \ldots, \boldsymbol{A}_m) \qquad \boldsymbol{C} ::= [\cdot] \qquad \boldsymbol{C} ::= \boldsymbol{C}_1[\boldsymbol{C}_2]$$

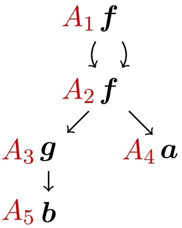
An STG is a **directed acyclic graph (DAG)**, if $\mathcal{CN} = \emptyset$.

Examples

DAGs allow **sharing** of **subtrees**



$\Longrightarrow$



$$
\begin{array}{lcl}
A_1 & ::= & f(A_2, A_2) \\
A_2 & ::= & f(A_3, A_4) \\
A_3 & ::= & g(A_5) \\
A_4 & ::= & a \\
A_5 & ::= & b
\end{array}
$$

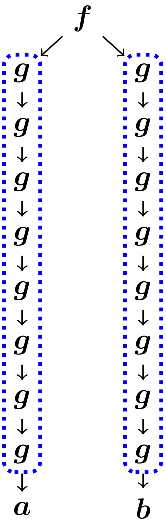
Examples

STGs additionally allow **sharing** and **compression** of **contexts**

$$
\begin{tikzcd}
& f \arrow[dl] \arrow[dr] & \\
g \arrow[d] & & g \arrow[d] \\
g \arrow[d] & & g \arrow[d] \\
g \arrow[d] & & g \arrow[d] \\
g \arrow[d] & & g \arrow[d] \\
g \arrow[d] & & g \arrow[d] \\
g \arrow[d] & & g \arrow[d] \\
g \arrow[d] & & g \arrow[d] \\
g \arrow[d] & & g \arrow[d] \\
a & & b
\end{tikzcd}
$$

Examples

STGs additionally allow **sharing** and **compression** of **contexts**



Examples

STGs additionally allow **sharing** and **compression** of **contexts**

$$
\begin{tikzcd}
 & f \arrow[dl] \arrow[dr] & \\
C_8 \arrow[d] & & C_8 \arrow[d] \\
a & & b
\end{tikzcd}
\qquad\qquad
C_8: \quad
\begin{tikzcd}
g \arrow[d] \\ g \arrow[d] \\ g \arrow[d] \\ g \arrow[d] \\ g \arrow[d] \\ g \arrow[d] \\ g \arrow[d] \\ g \arrow[d] \\ {[\cdot]}
\end{tikzcd}
$$

Examples

STGs additionally allow **sharing** and **compression** of **contexts**

$$\begin{array}{ccc} & f & \\ \swarrow & & \searrow \\ C_8 & & C_8 \\ \downarrow & & \downarrow \\ a & & b \end{array}$$

C_8 :

$$\begin{array}{c} g \\ \downarrow \\ g \\ \downarrow \\ g \\ \downarrow \\ g \\ \downarrow \\ g \\ \downarrow \\ g \\ \downarrow \\ g \\ \downarrow \\ g \\ \downarrow \\ [\cdot] \end{array}$$

Examples

STGs additionally allow **sharing** and **compression** of **contexts**

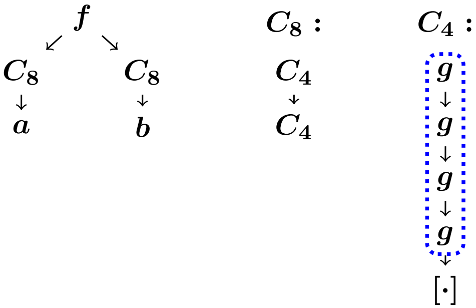
$$\begin{tikzcd}[column sep=small] & f \arrow[dl] \arrow[dr] & \\ C_8 \arrow[d] & & C_8 \arrow[d] \\ a & & b \end{tikzcd}$$

$$C_8 : \quad \begin{tikzcd} C_4 \arrow[d] \\ C_4 \end{tikzcd}$$

$$C_4 : \quad \begin{tikzcd} g \arrow[d] \\ g \arrow[d] \\ g \arrow[d] \\ g \arrow[d] \\ [\cdot] \end{tikzcd}$$

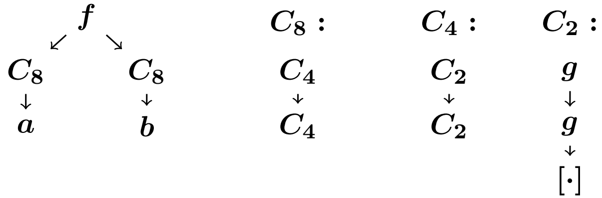
Examples

STGs additionally allow **sharing** and **compression** of **contexts**



Examples

STGs additionally allow **sharing** and **compression** of **contexts**



$$\begin{array}{lcl} A_1 & ::= & f(A_2, A_3) \\ A_2 & ::= & C_8[A_4] \\ A_3 & ::= & C_8[A_5] \\ A_4 & ::= & a \\ A_5 & ::= & b \end{array} \qquad \begin{array}{lcl} C_1 & ::= & g([\cdot]) \\ C_2 & ::= & g[C_1] \\ C_4 & ::= & C_2[C_2] \\ C_8 & ::= & C_4[C_4] \end{array}$$

Compression and Equality Check

Size $|G|$ of STG G = sum of sizes of all rhs of all production rules

$\mathrm{val}_G(A)$ = term generated by nonterminal A in grammar G

For every STG: term size and depth of $\mathrm{val}_G(A) = O(2^{|G|})$

Example

$$
\begin{array}{lcll}
A & ::= & C_n[A_a] & \mathrm{val}(A) = f^{2^n}(a) \\
C_{i+1} & ::= & C_i[C_i] \text{ for } i = 1, \ldots, n & \mathrm{val}(C_i) = f^{2^i}([\cdot]) \\
C_0 & ::= & f([\cdot]) & \mathrm{val}(C_0) = f([\cdot]) \\
A_a & = & a & \mathrm{val}(A_a) = a
\end{array}
$$

Proposition (Busatto, Lohrey, Maneth '05; Plandowski '94; Lifshits '07)

For an STG G and two term nonterminals A_1, A_2 it can be decided in $O(|G|^3)$ whether $\mathrm{val}_G(A_1) = \mathrm{val}_G(A_2)$ holds.

The E-Word Problem

- Given a set ground equations $E = \{u_1 = v_1, \ldots, u_n = v_n\}$
- Let $=_E$ be the smallest congruence on terms satisfying E

For ground terms s_1, s_2 the E-**word problem** is the question whether $s_1 =_E s_2$ holds.

We analyze this problem under compression:

- E and s_1, s_2 **DAG**-compressed: decidable in time $O(n \log n)$ by computing the congruence closure where n is the size of the input
- E and s_1, s_2 **STG**-compressed: obviously in DEXPTIME, exact lower bound unknown
- E **DAG**-compressed, s_1, s_2 **STG**-compressed: our **main result**: decidable in **polynomial time**

E-word problem: STG-compr. Terms, DAG-compr. Equations

Algorithm

Input: – Ground equations $L_1 = R_1, \dots, L_n = R_n$ where L_i, R_i are nonterminals of DAG G_E,
– Nonterminals S_1, S_2 of STG G_{Inp} representing terms s_1, s_2

Output: Yes or No ($s_1 =_E s_2$)

1. Compute a DAG G_T that represents a reduced ground TRS T which is equivalent to G_E using Snyder's algorithm (Snyder '89, '93)
2. Optimally compress the DAG G_T (Kozen' 77; Shostak '78; Nelson & Oppen '80)
3. Construct an STG G' that represents the STG-compressed T-normal forms of all term nonterminals of G_{Inp}
4. Use the Plandowski-Lifshits algorithm to decide whether S_1, S_2 represent the same terms.

E-word problem: STG-compr. Terms, DAG-compr. Equations

GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Algorithm

Input: – Ground equations $L_1 = R_1, \ldots, L_n = R_n$ where L_i, R_i are nonterminals of DAG G_E,

– Nonterminals S_1, S_2 of STG G_{Inp} representing terms s_1, s_2

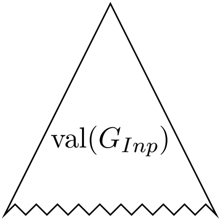
Output: Yes or No ($s_1 =_E s_2$)

1. Compute a DAG G_T that represents a reduced ground TRS T which is equivalent to G_E using Snyder's algorithm (Snyder '89, '93)
2. Optimally compress the DAG G_T (Kozen' 77; Shostak '78; Nelson & Oppen '80)
3. Construct an STG G' that represents the STG-compressed T-normal forms of all term nonterminals of G_{Inp}
4. Use the Plandowski-Lifshits algorithm to decide whether S_1, S_2 represent the same terms.

Normalform Computation

Input: – Reduced ground TRS T as DAG G_T with nonterminals $L_i \rightarrow R_i$,
– STG G_{Inp}

Output: Compute the T-normalforms of all terms represented by G_{Inp}

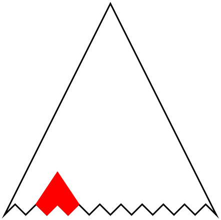


- Reduced & ground TRS
  $\Rightarrow$ normalization can be performed bottom up, since every contractum is irreducible

Normalform Computation

Input: – Reduced ground TRS T as DAG G_T with nonterminals $L_i \rightarrow R_i$,
– STG G_{Inp}

Output: Compute the T-normalforms of all terms represented by G_{Inp}

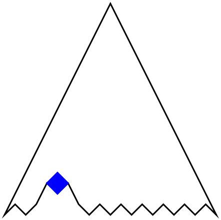


- Reduced & ground TRS
  $\Rightarrow$ normalization can be performed bottom up, since every contractum is irreducible

Normalform Computation

Input: – Reduced ground TRS T as DAG G_T with nonterminals $L_i \rightarrow R_i$,
– STG G_{Inp}

Output: Compute the T-normalforms of all terms represented by G_{Inp}

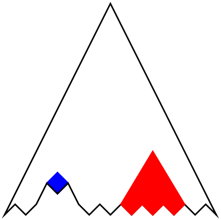


- Reduced & ground TRS
  $\Rightarrow$ normalization can be performed bottom up, since every contractum is irreducible

Normalform Computation

Input: – Reduced ground TRS T as DAG G_T with nonterminals $L_i \rightarrow R_i$,
– STG G_{Inp}

Output: Compute the T-normalforms of all terms represented by G_{Inp}

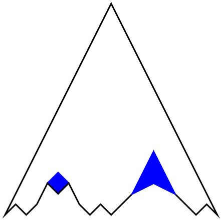


- Reduced & ground TRS
  $\Rightarrow$ normalization can be performed bottom up, since every contractum is irreducible

Normalform Computation

Input: – Reduced ground TRS T as DAG G_T with nonterminals $L_i \to R_i$,
– STG G_{Inp}

Output: Compute the T-normalforms of all terms represented by G_{Inp}

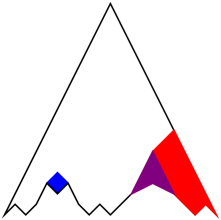


- Reduced & ground TRS
  $\Rightarrow$ normalization can be performed bottom up, since every contractum is irreducible

Normalform Computation

Input: – Reduced ground TRS T as DAG G_T with nonterminals $L_i \rightarrow R_i$,
– STG G_{Inp}

Output: Compute the T-normalforms of all terms represented by G_{Inp}

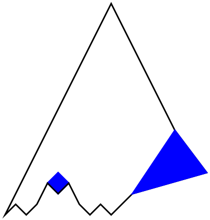


- Reduced & ground TRS
  $\Rightarrow$ normalization can be performed bottom up, since every contractum is irreducible

Normalform Computation

Input: – Reduced ground TRS T as DAG G_T with nonterminals $L_i \to R_i$,
– STG G_{Inp}

Output: Compute the T-normalforms of all terms represented by G_{Inp}

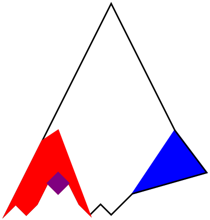


- Reduced & ground TRS
  $\Rightarrow$ normalization can be performed bottom up, since every contractum is irreducible

Normalform Computation

Input: – Reduced ground TRS T as DAG G_T with nonterminals $L_i \to R_i$,
– STG G_{Inp}

Output: Compute the T-normalforms of all terms represented by G_{Inp}

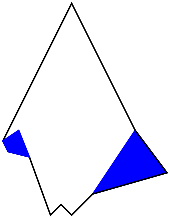


- Reduced & ground TRS
  $\Rightarrow$ normalization can be performed bottom up, since every contractum is irreducible

Normalform Computation

Input: – Reduced ground TRS T as DAG G_T with nonterminals $L_i \rightarrow R_i$,
– STG G_{Inp}

Output: Compute the T-normalforms of all terms represented by G_{Inp}

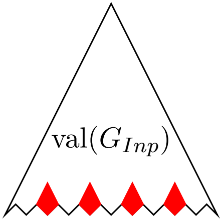


- Reduced & ground TRS
  $\Rightarrow$ normalization can be performed bottom up, since every contractum is irreducible

Normalform Computation

Input: – Reduced ground TRS T as DAG G_T with nonterminals $L_i \to R_i$,
– STG G_{Inp}

Output: Compute the T-normalforms of all terms represented by G_{Inp}

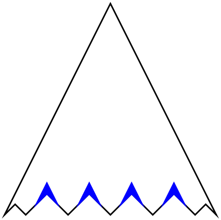


- Reduced & ground TRS
  $\Rightarrow$ normalization can be performed bottom up, since every contractum is irreducible
- $\mathrm{val}(G_{Inp})$ may have identical redexes at exponential many positions
- But the STG-representation shares the positions

Normalform Computation

Input: – Reduced ground TRS T as DAG G_T with nonterminals $L_i \to R_i$,
– STG G_{Inp}

Output: Compute the T-normalforms of all terms represented by G_{Inp}

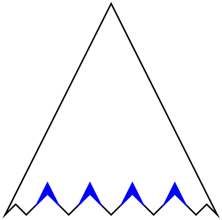


- Reduced & ground TRS
  $\Rightarrow$ normalization can be performed bottom up, since every contractum is irreducible
- $\mathrm{val}(G_{Inp})$ may have identical redexes at exponential many positions
- But the STG-representation shares the positions
- Normalization can also be “shared”, like a parallel rewriting step

Normalform Computation

Input: – Reduced ground TRS T as DAG G_T with nonterminals $L_i \to R_i$,
– STG G_{Inp}

Output: Compute the T-normalforms of all terms represented by G_{Inp}



- Reduced & ground TRS
  $\Rightarrow$ normalization can be performed bottom up, since every contractum is irreducible
- $\text{val}(G_{Inp})$ may have identical redexes at exponential many positions
- But the STG-representation shares the positions
- Normalization can also be "shared", like a parallel rewriting step

Algorithm has two phases:

Phase 1: Compute tables $\phi_\mathbf{0}, \phi_\mathbf{1}$ for the normalforms of almost all nonterminals by dynamic programming

Phase 2: Use $\phi_\mathbf{0}, \phi_\mathbf{1}$ to "normalize" G_{Inp}

Phase 1: ϕ-Computation

$$\mathrm{subterms}(G_T) = \bigcup_i \{A \mid L_i \xrightarrow{+}_{G_T} A\} \cup \bigcup_i \{R_i\} \cup \{\top\}$$

"$\top$ and all nodes of G_T without L_i and proper subterms of R_i"

Compute two tables bottom up along the grammar G_{Inp}

- For every **term nonterminal** A of G_{Inp}: $\boldsymbol{\phi_0}(A) \in \mathrm{subterms}(G_T)$
  - $\boldsymbol{\phi_0}(A) = N$, if $\mathrm{val}(N) = \mathsf{nf}_T(\mathrm{val}(A))$
  - $\boldsymbol{\phi_0}(A) = \top$, otherwise
- For every **context nonterminal** C of G_{Inp}: $\boldsymbol{\phi_1}(C) :: \mathrm{subterms}(G_T) \to \mathrm{subterms}(G_T)$ represents the mapping behavior of C on $\mathrm{subterms}(G_T)$ after normalization

Informally: If $\phi_0(A) = \top$, then normalization stops above A

ϕ-Computation

Computing $\phi_\mathbf{0}(A)$ for $A ::= f(A_1, \ldots, A_n)$

- "Normalize all subterms of A", i.e. compute $f(\phi_\mathbf{0}(A_1), \ldots, \phi_\mathbf{0}(A_n))$
- Does exist a production $N ::= f(\phi_\mathbf{0}(A_1), \ldots, \phi_\mathbf{0}(A_n))$ in G_T?
- If $N = L_i$, then $\phi_\mathbf{0}(A) = R_i$ (found a redex)
- If $N \in \mathrm{subterms}(G_T)$ but $N \neq L_i$, then $\phi_\mathbf{0}(A) = N$
  ($f(\phi_\mathbf{0}(A_1), \ldots, \phi_\mathbf{0}(A_n))$ maybe a subterm of a redex)
- Otherwise, $\phi_\mathbf{0}(A) = \top$
  ($f(\phi_\mathbf{0}(A_1), \ldots, \phi_\mathbf{0}(A_n))$ not a redex and not a subterm of a redex)

ϕ-Computation (2)

Other cases

- $C ::= f(A_1, \dots, [\cdot], \dots, A_n)$:
  compute $f(\boldsymbol{\phi_0}(A_1), \dots, X, \dots, \boldsymbol{\phi_0}(A_n))$ for any $X \in \mathrm{subterms}(G_T)$
  expensive case, requires time $O(|G_T| \cdot \log |G_T|)$
- $C ::= C_1[C_2]$ then $\boldsymbol{\phi_1}(C) = \boldsymbol{\phi_1}(C) \circ \boldsymbol{\phi_1}(C)$
- $C ::= [\cdot]$ then $\boldsymbol{\phi_1}(C) = Id$
- $A ::= B$ then $\boldsymbol{\phi_0}(A) = \boldsymbol{\phi_0}(B)$
- $A ::= C[B]$ then $\boldsymbol{\phi_0}(A) = \boldsymbol{\phi_1}(C)(\boldsymbol{\phi_0}(B))$

Phase 2: Normalization using ϕ

- If $\boldsymbol{\phi_0}(A) = \top$ then $\mathrm{val}(A)$ is not a redex, and every superterm of $\mathrm{val}(A)$ is not a redex. This also holds after reducing inside $\mathrm{val}(A)$.
- Otherwise $\boldsymbol{\phi_0}(A)$ is the normal form of A

Phase 2: Normalization using ϕ

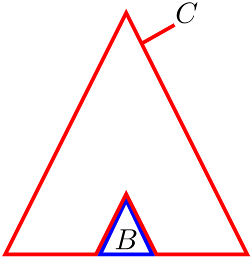
- If $\phi_{\mathbf{0}}(A) = \top$ then $\mathrm{val}(A)$ is not a redex, and every superterm of $\mathrm{val}(A)$ is not a redex. This also holds after reducing inside $\mathrm{val}(A)$.
- Otherwise $\phi_{\mathbf{0}}(A)$ is the normal form of A

$\Rightarrow$ **Normalization:** Modify G_{Inp} (rules for all $C \in \mathcal{CN}$ unchanged):

- If $\phi_{\mathbf{0}}(A) = N$ then replace rule for A by $A ::= N$
- If $\phi_{\mathbf{0}}(A) = \top$ then rule is unchanged, except for:

  $A ::= C[B]$ and $\phi_{\mathbf{0}}(B) = N \neq \top$:

  - **Split** C into $C_1[C_2]$ using the grammar s.t.
  - $\phi_{\mathbf{1}}(C_2)(N) \neq \top$ and C_2 is maximal.
  - Replace rule by $A ::= C_1[\phi_{\mathbf{1}}(C_2)(N)]$.
  - Productions for C_1 may **increase the size** of the grammar by $O(|G_{Inp}|)$



Phase 2: Normalization using ϕ

- If $\boldsymbol{\phi_0}(A) = \top$ then $\mathrm{val}(A)$ is not a redex, and every superterm of $\mathrm{val}(A)$ is not a redex. This also holds after reducing inside $\mathrm{val}(A)$.
- Otherwise $\boldsymbol{\phi_0}(A)$ is the normal form of A

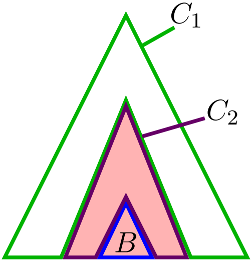
$\Rightarrow$ **Normalization:** Modify G_{Inp} (rules for all $C \in \mathcal{CN}$ unchanged):

- If $\boldsymbol{\phi_0}(A) = N$ then replace rule for A by $A ::= N$
- If $\boldsymbol{\phi_0}(A) = \top$ then rule is unchanged, except for:

  $A ::= C[B]$ and $\boldsymbol{\phi_0}(B) = N \neq \top$:

  - **Split** C into $C_1[C_2]$ using the grammar s.t.
  - $\boldsymbol{\phi_1}(C_2)(N) \neq \top$ and C_2 is maximal.
  - Replace rule by $A ::= C_1[\boldsymbol{\phi_1}(C_2)(N)]$.

  - Productions for C_1 may **increase the size** of the grammar by $O(|G_{Inp}|)$



Phase 2: Normalization using ϕ

- If $\boldsymbol{\phi_0}(A) = \top$ then $\mathrm{val}(A)$ is not a redex, and every superterm of $\mathrm{val}(A)$ is not a redex. This also holds after reducing inside $\mathrm{val}(A)$.
- Otherwise $\boldsymbol{\phi_0}(A)$ is the normal form of A

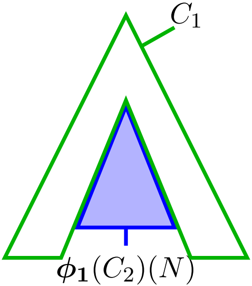
$\Rightarrow$ **Normalization:** Modify G_{Inp} (rules for all $C \in \mathcal{CN}$ unchanged):

- If $\boldsymbol{\phi_0}(A) = N$ then replace rule for A by $A ::= N$
- If $\boldsymbol{\phi_0}(A) = \top$ then rule is unchanged, except for:

  $A ::= C[B]$ and $\boldsymbol{\phi_0}(B) = N \neq \top$:

  - **Split** C into $C_1[C_2]$ using the grammar s.t.
  - $\boldsymbol{\phi_1}(C_2)(N) \neq \top$ and C_2 is maximal.
  - Replace rule by $A ::= C_1[\boldsymbol{\phi_1}(C_2)(N)]$.

  - Productions for C_1 may **increase the size** of the grammar by $O(|G_{Inp}|)$



Complexity

Let $G := G_{Inp} \cup G_E$

1. Compute a DAG G_T that represents a reduced ground TRS T which is equivalent to G_E using Snyder's algorithm
   time: $O(|G_E| \cdot \log^2 |G_E|)$, space $|G_T| = O(|G_E|)$
2. Optimally compress the DAG G_T
   time: $O(|G_E \cdot \log |G_E|)$
3. Construct an STG G' that represents the STG-compressed T-normal forms of all term nonterminals of G_{Inp}
   time:$O(\underbrace{|G|^2}_{\text{Normalization}} + \underbrace{O(|G_{Inp}| \cdot |G_T| \cdot \log(|G_T|))}_{\phi\text{-computation}})$, space $\underbrace{|G'| = O(|G|^2)}_{\text{Normalization}}$
4. Use the Plandowski-Lifshits algorithm to decide whether S_1, S_2 represent the same terms.
   time: $O(|G'|^3) = O(|G|^6)$

STG-Compressed Equations

If equations E (grammar G_E,resp.) and s, t are STG-compressed: Exact lower bound unknown.

We considered STG-compressed ground TRS G_T and normalization:

- Normalization is NP-hard.
  Proof is an encoding of positive SUBSETSUM
- Normalization is in PSPACE.
  Proof: For a reduction sequence $s_1 \to s_2 \ldots \to s_n = \mathsf{nf}_T(s_1)$ show: Every grammar corresponding to s_i can be represented in polynomial space.

$\Rightarrow$ Using normalization does not efficiently work for the STG-compressed case

Conclusion

- E-word problem is efficiently decidable for DAG-compressed E, STG-compressed input
- We implemented a prototype in about 2000 lines of Haskell code
- E-word problem for STG-compressed E requires other methods
- But note: Usually the equations E are much smaller than the input terms

Future Work:

- Find a good lower bound for STG-compressed E
- Other open problems for the compressed case: non-ground TRS, completion, etc.