Decision Methods for Concurrent Kleene Algebra with Tests : Based on Derivative

Yoshiki Nakamura

Tokyo Institute of Technology

RAMiCS2015 September 28, 2015
abstract

- Concurrent Kleene Algebra with Tests (CKAT) is introduced in RAMiCS2014 [Jipsen 2014]
- We give decision methods for CKAT (based on Derivative).
- Additionally, considering the computational complexity of CKAT (in EXPSPACE)
CKAT

- CKAT is Kleene Algebra (KA) with
  - Boolean Test (derived from KAT [Kozen and Smith 1996])
  - Concurrent Operator $\parallel$ (related to Concurrent KA [Hoare et al. 2009])
- Each CKAT term is an expression of *guarded series-parallel language*. 
Guarded series-parallel language (gsp language)

**Definition (gsp string)**

The gsp strings set is a smallest set s.t.

- $\alpha$ is a gsp string
- $\alpha_1 p \alpha_2$ is a gsp string
- If $w_1$ and $w_2$ are gsp strings, then $w_1 \diamond w_2$ is a gsp string
- If $w_1$ and $w_2$ are gsp strings, then $w_1 \parallel w_2$ is a gsp string

Where $\alpha$ is a subset of basic tests and $p$ is a basic programs.

**Concatenation**

$$w_1 \alpha \diamond \alpha' w_2 = \begin{cases} w_1 \alpha w_2 & (\alpha = \alpha') \\ \text{undefined} & (\text{o.w.}) \end{cases}$$

**Parallel Composition**

$$w_1 \parallel w_2 = \begin{cases} \alpha_1 \{|w'_1, w'_2|\} \alpha_2 & (w_1 = \alpha_1 w'_1 \alpha_2, w_2 = \alpha_1 w'_2 \alpha_2) \\ \alpha & (w_1 = w_2 = \alpha) \\ \text{undefined} & (\text{o.w.}) \end{cases}$$
gsp language

**definition (gsp language** \( L(p) \))

- \( L(p_1 \parallel p_2) = \{ \alpha_1\{w_1, w_2\}\alpha_2 \mid \alpha_1w_1\alpha_2 \in L(p_1), \alpha_1w_2\alpha_2 \in L(p_2) \} \)

(the other cases)

- \( L(b) = \{ \alpha \mid b \in \alpha \} \) for any boolean term \( b \)
- \( L(p) = \{ \alpha_1p\alpha_2 \mid \alpha_1, \alpha_2 \) are the subset of basic tests\} \)
- \( L(p_1 + p_2) = L(p_1) \cup L(p_2) \)
- \( L(p_1p_2) = \{ \alpha_1w_1\alpha_2w_3\alpha_3 \mid \alpha_1w_1\alpha_2 \in L(p_1) \) and \( \alpha_2w_2\alpha_3 \in L(p_1) \} \)
- \( L(p^*) = \bigcup_{n<\omega} \{ \alpha_0w_1\alpha_1 \ldots w_n\alpha_n \mid \alpha_{i-1}w_i\alpha_i \in L(p) \} \)
- \( (L(P) = \bigcup_{p \in P} L(p) \) for any CKAT term set \( P) \)

**example:** \( T = \{ t_1, t_2 \} \)

- \( L(t_1a \parallel t_2b t_1) = \{ T\{a, b\}\}T, T\{a, b\}\{t_1\} \}
- \( L(t_1a \parallel t_1) = \emptyset \)
- \( L(t_1 \parallel t_1) = \{ \{t_1\}, T \} \)
Derivative

- Derivative is first introduced by Brzozowski [Brzozowski 1964] for Kleene Algebra.
- Derivative has many applications to many language theoretic problems, for example
  - membership problem
  - emptiness problem
  - equivalence problem
  - ... and so on
- Derivative $D_w$ is aimed to satisfy $w^{-1}L(p) = L(D_w(p))$.
  - $w^{-1}$ is a left quotient by $w$.
  - $w^{-1}L(p) = \{w' \mid w \diamond w' \in L(p)\}$
  - e.g. $(TaT)^{-1}\{TaT, TaTbT, TbT, TbTbT\} = \{T, TbT\}$
- We give the derivative for CKAT in the next page.
### Naive Derivative for CKAT

**Definition (Naive Derivative \( D_w(p) \))**

- \( D_{\alpha \mid w_1, w_2} \alpha'(p_1 \parallel p_2) = \emptyset \)
- \( D_{\alpha \{ |w_1, w_2| \}} \alpha'(p) = \emptyset \)
- \( D_{\alpha \{ |w_1, w_2| \}} \alpha'(p_1 \parallel p_2) = E_{\alpha'}((D_{\alpha w_1} \alpha'(p_1) \parallel D_{\alpha w_2} \alpha'(p_2)) \cup (D_{\alpha w_2} \alpha'(p_1) \parallel D_{\alpha w_1} \alpha'(p_2))) \)

*(the other cases)*

- \( D_{\alpha w \alpha' \mid w' \alpha''}(p) = (D_{\alpha w} \alpha' \circ D_{\alpha' w'} \alpha'')(p) \)
- \( D_{\alpha w} \alpha'(p_1 + p_2) = D_{\alpha w} \alpha'(p_1) \cup D_{\alpha w} \alpha'(p_2) \)
- \( D_{\alpha w} \alpha'(p_1 p_2) = D_{\alpha w} \alpha'(p_1) \{ p_2 \} \cup E_{\alpha}(p_1)D_{\alpha w} \alpha'(p_2) \)
- \( D_{\alpha w} \alpha'(p_1^*) = D_{\alpha w} \alpha'(p_1) \{ p_1^* \} \)
- \( D_{\alpha w} \alpha'(b) = \emptyset \) for any boolean term \( b \)

Where \( E_{\alpha}(p_1) = \begin{cases} \{1\} & (\alpha \in L(p_1)) \\ \emptyset & (o.w) \end{cases} \).
Naive Derivative for CKAT

**Theorem**

For any gsp string $w\alpha'$ and CKAT term $p$,

$$(w\alpha')^{-1}L(p) = \alpha'^{-1}L(D_w\alpha'(p))$$

- We only check whether $\{1\} = E_{\alpha'}(D_{\alpha w\alpha'}(p))$ or not to decide $w \in L(p)$.

But, this derivative need too many spaces.

- The enough length of string to decide many language problems is too large. (e.g. In KA, $2^p(\text{input size})$.)
- We need to memorize $w_1$ and $w_2$ (too large!) to calculate $D_{\alpha\{|w_1,w_2|\}}\alpha'(p)$.
  
  $D_{\alpha\{|w_1,w_2|\}}\alpha'(p_1 \parallel p_2) = E_{\alpha'}((D_{\alpha w_1\alpha'}(p_1) \parallel D_{\alpha w_2\alpha'}(p_2)) \cup (D_{\alpha w_2\alpha'}(p_1) \parallel D_{\alpha w_1\alpha'}(p_2)))$
Naive Derivative for CKAT

**Theorem**

For any gsp string $w_\alpha'$ and CKAT term $p$, 
$$(w_\alpha')^{-1}L(p) = \alpha'^{-1}L(D_{w_\alpha'}(p))$$

- We only check whether $\{1\} = E_{\alpha'}(D_{\alpha w_\alpha'}(p))$ or not to decide $w \in L(p)$.

But, this derivative need too many spaces.

- The enough length of string to decide many language problems is too large. (e.g. In KA, $2^p$(input size).)
- We need to memorize $w_1$ and $w_2$(too large!) to calculate $D_{\alpha\{|w_1,w_2|\}\alpha'}(p)$.
  
  $D_{\alpha\{|w_1,w_2|\}\alpha'}(p_1 \parallel p_2) = E_{\alpha'}((D_{\alpha w_1\alpha'}(p_1) \parallel D_{\alpha w_2\alpha'}(p_2)) \cup (D_{\alpha w_2\alpha'}(p_1) \parallel D_{\alpha w_1\alpha'}(p_2)))$

So, we want to get more efficient derivative not to memorize long strings.
Outline

For example, when \( w = \alpha\{|p\alpha p', q|\} \alpha r \alpha' \)

- **Naive Derivative**
  \[
  \frac{\alpha p}{\alpha p} \rightarrow \frac{\alpha p'}{\alpha p} \rightarrow \frac{\alpha p\alpha p'}{\alpha p}
  \]

  \[
  \alpha\{|p\alpha p', q|\} \xrightarrow{\alpha r} \alpha\{|p\alpha p', q|\} \alpha r
  \]

  \[
  \frac{\alpha q}{\alpha q} \rightarrow \frac{\alpha q}{\alpha q}
  \]

- **Memory Efficient Derivative**
  \[
  \frac{\alpha\{|p, q|}{\alpha\{|p, q|} \rightarrow \frac{\alpha p'}{\alpha\{|p, q|} \rightarrow \frac{\alpha\{|p\alpha p', q|}{\alpha p'} \rightarrow \frac{\alpha\{|p\alpha p', q|}{\alpha r}
  \]
Outline

Memory Efficient Derivative

\[ \alpha\{p, q\} \xrightarrow{\alpha p'} \alpha\{p \alpha p', q\} \xrightarrow{\{\} \alpha r} \alpha\{p \alpha p', q\}\alpha r \]

- We can forget the gray part to calculate.

But, this derivative has no uniqueness. When \( \alpha p' \) is inputed, we cannot decide whether

- \( \alpha\{p, q\} \xrightarrow{\alpha p'} \alpha\{p \alpha p', q\} \) or
- \( \alpha\{p, q\} \xrightarrow{\alpha p'} \alpha\{p, q \alpha p'\} \)

To distinguish them, we introduce derivative variables.

- \( \alpha\{p x, q y\} \xrightarrow{x + = \alpha p'} \alpha\{p \alpha p' x, q y\} \)
- \( \alpha\{p x, q y\} \xrightarrow{y + = \alpha p'} \alpha\{p x, q \alpha p' y\} \)
Memory Efficient Derivative

To express derivative variables, we expand CKAT terms to intermediate CKAT terms to add $D_x(p)$. For example, when $p = (pp' \parallel q); r$, 

$$D_x((pp' \parallel q); r) \xrightarrow{x + = \alpha\{py, qz\}} D_x((D_y(p') \parallel D_z(1)); r)$$

$$D_x((D_y(1) \parallel D_z(1)); r) \xrightarrow{x + = \alpha r} D_x(1)$$

$$y + = \alpha p'$$
Intermediate CKAT term and Memory Efficient Derivative

We introduce the new derivative functions $D_{x+\alpha\mathcal{T}}$ for Intermediate CKAT terms. ($\mathcal{T} := p \mid \{\mathcal{T}_1x_1, \mathcal{T}_2x_2\}$)

definition (Memory Efficient Derivative $D_{x+\alpha\mathcal{T}}$)

- $E_\alpha(D_x(p)) = E_\alpha(p)$
- $D_{x+\alpha\mathcal{T}}(D_x(p)) = D_x(\text{join}_\alpha \circ D_{\alpha\mathcal{T}}(p))$
- $\text{join}_\alpha(D_x(p)) = E_\alpha(p)$
- $D_{\alpha\{\mathcal{T}_1x_1, \mathcal{T}_2x_2\}}(p_1 \parallel p_2) = D_{x_1}(D_{\alpha\mathcal{T}_1}(p_1)) \parallel D_{x_2}(D_{\alpha\mathcal{T}_2}(p_2)) \cup D_{x_2}(D_{\alpha\mathcal{T}_2}(p_1)) \parallel D_{x_1}(D_{\alpha\mathcal{T}_1}(p_2))$

In the other cases of the above definitions, they take no actions. (More precisely, it means as follows)

- $D_{x+\alpha\mathcal{T}}(p + q) = D_{x+\alpha\mathcal{T}}(p) + D_{x+\alpha\mathcal{T}}(q)$
- $D_{x+\alpha\mathcal{T}}(D_y(p)) = D_y(D_{x+\alpha\mathcal{T}}(p))$ for $y \neq x$
- $D_{x+\alpha\mathcal{T}}(p) = p$ for any basic test $p$
- $\ldots$
Naive Derivative and Memory Efficient Derivative

**Theorem**

\[ D_{w\alpha'} \circ E_{\alpha'}(p) = D_{x+w} \circ E_{\alpha'}(D_x(p)) \]

- Naive Derivative can be replaced to Memory Efficient Derivative.

We next consider the computational complexity of Memory Efficient Derivative.
After this, in particular, we consider the equivalence problem of CKAT.
Outline of the computational complexity

- The intermediate CKAT terms by memory efficient derivative are in a closure.
- The closure size is bounded.
- When two CKAT terms are not equivalent, there exists a gsp string (witness) whose intersection width is less than the max intersection width of them.

Note that intersection width of CKAT term $p \ iw(p)$ and Intersection width of gsp strings $w \ iw(w)$ are defined, respectively.

**examples**

- $iw(((p || q) || r)(p || q))) = 3$
- $iw((T{|a,b|}Tc)) = 2$
Closure

We define the closure $C{l_X}(p)$, where $p$ is a intermediate CKAT term and $X$ is a set of derivative variables, as follows,

definition (Closure $C{l_X}(p)$)

- $C{l_X}(a) = \{a\}$ for $a = 0 \mid 1 \mid t$
- $C{l_X}(\overline{b}) = \{\overline{b}\} \cup C{l_X}(b)$ for any boolean term $b$
- $C{l_X}(p) = \{p, 1\}$
- $C{l_X}(q_1 + q_2) = \{q_1 + q_2\} \cup C{l_X}(q_1) \cup C{l_X}(q_2)$
- $C{l_X}(q_1q_2) = \{q_1q_2\} \cup C{l_X}(q_1)\{q_2\} \cup C{l_X}(q_2)$
- $C{l_X}(q_1^*) = \{q_1^*\} \cup C{l_X}(q_1)\{q_1^*\}$
- $C{l_X}(q_1 \parallel q_2) = \{q_1 \parallel q_2\} \cup \{D_{x_1}(q_1') \parallel D_{x_2}(q_2') \mid (q_1' \in C{l_X}(q_1), q_2' \in C{l_X}(q_2))$ or $(q_1' \in C{l_X}(q_2), q_2' \in C{l_X}(q_1)), x_1, x_2 \in X\}$
- $C{l_X}(D_x(q_1)) = \{D_x(q_1)\} \cup D_x(C{l_X}(q_1))$
Closure

**Theorem**

If any derivative variables occurred in $q$ are in $X$,

$$D_{x^+} = \alpha T(q) \subseteq Cl_X(q)$$

Because $Cl_X$ is a closed operator,

$$D_{x_1^+} = \alpha_1 T_1 \circ \cdots \circ D_{x_n^+} = \alpha_n T_n(q) \subseteq Cl_X(q)$$

**Theorem**

$$|Cl_X(q)| \leq 2 \ast |q|^i w(q) \ast |X|^{2 \ast i w(q)}$$

where $i w(q)$ is the intersection width of $q$. 
Intersection width is bounded

**Theorem**

If $iw(w) > iw(q)$, $D_{x+=w}(q) = \emptyset$

- If $iw(w) > \max(iw(p), iw(q))$, then $iw(p) = iw(q) = \emptyset$.
  - We only consider the case of $iw(w) \leq \max(iw(p), iw(q))$.

- Let $IW = \max(iw(p), iw(q))$. Each intermediate CKAT term whose $iw$ is less than $IW$ has at most $2 \times IW - 1$ derivative variables.
  - We can assume $|X| \leq 2 \times IW - 1$.
  - $|Cl_{X}(q)| \leq 2 \times |q|^{iw(q)} \times |X|^{2 \times iw(q)} \leq 2 \times |q|^{IW} \times (2 \times IW - 1)^{2 \times IW}$
  - $|Cl_{X}(p)|, |Cl_{X}(q)| \leq 2 \times l^{l} \times (2 \times l - 1)^{2 \times l}$, where $l$ is $|p| + |q|$.
  - Therefore, the closure size is $O(2^{p(l)})$, where $p$ is a polynomial function of $l$. 
The equivalence problem of CKAT is in EXPSPACE.

(Outline of EXPSPACE algorithm)

- We nondeterministically select the syntax of $x + = \alpha T$ and rewrite $p$ and $q$ to $D_{x+ = \alpha T}(p)$ and $D_{x+ = \alpha T}(q)$, respectively.
  - We are enough to select $T$ s.t. $iw(T) \leq IW$.
  - By Savitch’s theorem [Savitch 1970], EXPSPACE = NEXPSPACE.
- During execution, if we find the case of $E_{\alpha'}(p) \neq E_{\alpha'}(q)$, then $p$ and $q$ is not equivalent.
  - The loop count of this algorithm is finite because the pattern of $(p, q)$ is at most $2|Cl_X(p)| \ast 2|Cl_X(q)| = O(2^{2p(l)})$, where $p$ is a polynomial function of $l$.
  - We only memorize $p$ and $q$ and the step count. these are enough to prepare exponential spaces because $|Cl_X(p)| = O(2^{p(l)})$ and $|Cl_X(q)| = O(2^{p(l)})$. 

N.Yoshiki (TokyoTech)
Fixed Parameter

**Theorem**
The equivalence problem of CKAT is in EXPSPACE.

**Corollary**
If the maximum of the intersection width is a fixed parameter, the equivalence problem of CKAT is PSPACE-complete.

(PSPACE-hardness is derived by [Hunt III 1973].)
Concluding Remarks

- concluding summary
  - We have given the derivative for CKAT.
  - We have shown that the equivalence problem of CKAT is in EXPSPACE.

- Future works
  - Is this equivalence problem EXPSPACE-complete?
  - If we allow $\epsilon$ (for example, $\alpha\{|p, \epsilon|\}\alpha$), can we give efficient derivative? (It become a little difficult because we have to memorize $\alpha$ in the case of $x += \alpha\{|p_1x_1, \epsilon|\}$. We should give another derivative to show the result like the corollary of PSPACE.)

This is all for my presentation.


bibliography II
