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

Yoshiki Nakamura

Tokyo Instutute 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 $\|$ (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.

- α is a gsp string
- $\alpha_1 \mathbf{p} \alpha_2$ 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 α is a subset of basic tests and $\mathbf{p}$ is a basic programs.

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

Parallel Composition

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

gsp language

definition (gsp language $L(p)$)

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

(the other cases)

- $L(b) = \{\alpha \mid b \in \alpha\}$ for any boolean term b
- $L(\mathbf{p}) = \{\alpha_1 \mathbf{p} \alpha_2 \mid \alpha_1, \alpha_2$ are the subset of basic tests$\}$
- $L(p_1 + p_2) = L(p_1) \cup L(p_2)$
- $L(p_1 p_2) = \{\alpha_1 w_1 \alpha_2 w_3 \alpha_3 \mid \alpha_1 w_1 \alpha_2 \in L(p_1)$ and $\alpha_2 w_2 \alpha_3 \in L(p_1)\}$
- $L(p^*) = \cup_{n<\omega}\{\alpha_0 w_1 \alpha_1 \ldots w_n \alpha_n \mid \alpha_{i-1} w_i \alpha_i \in L(p)\}$
- $(L(P) = \cup_{p \in P} L(p)$ for any CKAT term set $P)$

example: $T = \{\mathbf{t}_1, \mathbf{t}_2\}$

- $L(\mathbf{t}_1 a \parallel \mathbf{t}_2 b \mathbf{t}_1) = \{T\{|a, b|\}T, T\{|a, b|\}\{\mathbf{t}_1\}\}$
- $L(\mathbf{t}_1 a \parallel \mathbf{t}_1) = \emptyset$
- $L(\mathbf{t}_1 \parallel \mathbf{t}_1) = \{\{\mathbf{t}_1\}, T\}$

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 $D_w(p)$)

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

(the other cases)

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

Where $E_{\alpha}(p_1) = \begin{cases} \{\mathbf{1}\} & (\alpha \in L(p_1)) \\ \emptyset & (o.w) \end{cases}$.

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 $\{\mathbf{1}\} = E_{\alpha'}(D_{\alpha w\alpha'}(p))$ 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 $\{\mathbf{1}\} = E_{\alpha'}(D_{\alpha w\alpha'}(p))$ 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 $D_{\alpha\{|w_1,w_2|\}\alpha'}(p)$.
  - $D_{\alpha\{|w_1,w_2|\}\alpha'}(p_1 \parallel p_2) = E_{\alpha'}((D_{\alpha w_1\alpha'}(p_1) \parallel D_{\alpha w_2\alpha'}(p_2)) \cup (D_{\alpha w_2\alpha'}(p_1) \parallel D_{\alpha w_1\alpha'}(p_2)))$

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

Outline

For example, when $w = \alpha\{|\mathbf{p}\alpha\mathbf{p}', \mathbf{q}|\}\alpha\mathbf{r}\alpha'$

- Naive Derivative

$$\begin{tikzcd}
{} \arrow[r, "\alpha\mathbf{p}"] & \alpha\mathbf{p} \arrow[r, "\alpha\mathbf{p}'"] & \alpha\mathbf{p}\alpha\mathbf{p}' \arrow[dr] & & \\
 & & & \alpha\{|\mathbf{p}\alpha\mathbf{p}', \mathbf{q}|\} \arrow[r, "\alpha\mathbf{r}"] & \alpha\{|\mathbf{p}\alpha\mathbf{p}', \mathbf{q}|\}\alpha\mathbf{r} \\
 & {} \arrow[r, "\alpha\mathbf{q}"] & \alpha\mathbf{q} \arrow[ur] & &
\end{tikzcd}$$

- Memory Efficient Derivative

$$\xrightarrow{\alpha\{|\mathbf{p},\mathbf{q}} \alpha\{|\mathbf{p},\mathbf{q} \xrightarrow{\alpha\mathbf{p}'} \alpha\{|\mathbf{p}\alpha\mathbf{p}',\mathbf{q} \xrightarrow{|\}\alpha\mathbf{r}} \alpha\{|\mathbf{p}\alpha\mathbf{p}',\mathbf{q}|\}\alpha\mathbf{r}$$

Outline

Memory Efficient Derivative

$$\xrightarrow{\alpha\{|\mathbf{p},\mathbf{q}} \alpha\{|\mathbf{p},\mathbf{q} \xrightarrow{\alpha\mathbf{p}'} \alpha\{|\mathbf{p}\alpha\mathbf{p}',\mathbf{q} \xrightarrow{|\}\alpha\mathbf{r}} \alpha\{|\mathbf{p}\alpha\mathbf{p}',\mathbf{q}|\}\alpha\mathbf{r}$$

- We can forget the gray part to calculate.

But, this derivative has no uniqueness. When $\alpha\mathbf{p}'$ is inputed, we cannot decide whether

- $\alpha\{|\mathbf{p},\mathbf{q} \xrightarrow{\alpha\mathbf{p}'} \alpha\{|\mathbf{p}\alpha\mathbf{p}',\mathbf{q}$ or
- $\alpha\{|\mathbf{p},\mathbf{q} \xrightarrow{\alpha\mathbf{p}'} \alpha\{|\mathbf{p},\mathbf{q}\alpha\mathbf{p}'$

To distinguish them, we introduce *derivative variables*.

- $\alpha\{|\mathbf{p}x,\mathbf{q}y \xrightarrow{x{+}{=}\alpha\mathbf{p}'} \alpha\{|\mathbf{p}\alpha\mathbf{p}'x,\mathbf{q}y$
- $\alpha\{|\mathbf{p}x,\mathbf{q}y \xrightarrow{y{+}{=}\alpha\mathbf{p}'} \alpha\{|\mathbf{p}x,\mathbf{q}\alpha\mathbf{p}'y$

Memory Efficient Derivative

Memory Efficient Derivative

$$\begin{tikzcd}
 & x \arrow[r, "x \mathrel{+}= \alpha\{|\mathbf{p}y, \mathbf{q}z|\}"] & \alpha\{|\mathbf{p}y, \mathbf{q}z|\}x \arrow[dl, "y \mathrel{+}= \alpha\mathbf{p}'"'] & \\
 & \alpha\{|\mathbf{p}\alpha\mathbf{p}'y, \mathbf{q}z|\}x \arrow[r, "x \mathrel{+}= \alpha\mathbf{r}"] & \alpha\{|\mathbf{p}\alpha\mathbf{p}', \mathbf{q}|\}\alpha\mathbf{r}x &
\end{tikzcd}$$

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

$$\begin{tikzcd}
D_x((\mathbf{p}\mathbf{p}' \parallel \mathbf{q}); \mathbf{r}) \arrow[r, "x \mathrel{+}= \alpha\{|\mathbf{p}y, \mathbf{q}z|\}"] & D_x((D_y(\mathbf{p}') \parallel D_z(\mathbf{1})); \mathbf{r}) \arrow[dl, "y \mathrel{+}= \alpha\mathbf{p}'"'] \\
D_x((D_y(\mathbf{1}) \parallel D_z(\mathbf{1})); \mathbf{r}) \arrow[r, "x \mathrel{+}= \alpha r"] & D_x(\mathbf{1})
\end{tikzcd}$$

Intermediate CKAT term and Memory Efficient Derivative

We introduce the new derivative functions $D_{x+=\alpha\mathcal{T}}$ for Intermediate CKAT terms. $(\mathcal{T} := \mathbf{p} \mid \{|\mathcal{T}_1 x_1, \mathcal{T}_2 x_2|\})$

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

- $E_\alpha(D_x(p)) = E_\alpha(p)$
- $D_{x+=\alpha\mathcal{T}}(D_x(p)) = D_x(join_\alpha \circ D_{\alpha\mathcal{T}}(p))$
- $join_\alpha(D_x(p)) = E_\alpha(p)$
- $D_{\alpha\{|\mathcal{T}_1 x_1, \mathcal{T}_2 x_2|\}}(p_1 \parallel p_2) = D_{x_1}(D_{\alpha\mathcal{T}_1}(p_1)) \parallel D_{x_2}(D_{\alpha\mathcal{T}_2}(p_2)) \cup D_{x_2}(D_{\alpha\mathcal{T}_2}(p_1)) \parallel D_{x_1}(D_{\alpha\mathcal{T}_1}(p_2))$

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 $y \neq x$
- $D_{x+=\alpha\mathcal{T}}(\mathbf{p}) = \mathbf{p}$ for any basic test $\mathbf{p}$
- $\ldots$

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

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

Closure

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

definition (Closure $Cl_X(p)$)

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

Closure

theorem

If any derivative variables occurred in q are in X,
$D_{x+=\alpha\mathcal{T}}(q) \subseteq Cl_X(q)$

Because Cl_X is a closed operator,
$D_{x_1+=\alpha_1\mathcal{T}_1} \circ \cdots \circ D_{x_n+=\alpha_n\mathcal{T}_n}(q) \subseteq Cl_X(q)$

theorem

$|Cl_X(q)| \leq 2 * |q|^{iw(q)} * |X|^{2*iw(q)}$
where $iw(q)$ 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 $2 * IW - 1$ derivative variables.
  - We can assume $|X| \leq 2 * IW - 1$.
  - $|Cl_X(q)| \leq 2 * |q|^{iw(q)} * |X|^{2*iw(q)} \leq 2 * |q|^{IW} * (2 * IW - 1)^{2*IW}$
  - $|Cl_X(p)|, |Cl_X(q)| \leq 2 * l^l * (2 * l - 1)^{2*l}$, 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

theorem

The equivalence problem of CKAT is in EXPSPACE.

(Outline of EXPSPACE algorithm)

- We nondeterministically select the syntax of $x \mathrel{+}= \alpha\mathcal{T}$ and rewrite p and q to $D_{x+=\alpha\mathcal{T}}(p)$ and $D_{x+=\alpha\mathcal{T}}(q)$, respectively.
  - We are enough to select $\mathcal{T}$ s.t. $iw(\mathcal{T}) \leq IW$.
  - 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 $2^{|Cl_X(p)|} * 2^{|Cl_X(q)|} = O(2^{2^{p(l)}})$, 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 $|Cl_X(q)| = O(2^{p(l)})$.

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 ϵ (for example, $\alpha\{|\mathbf{p}, \epsilon|\}\alpha$), can we give efficient derivative? (It become a little difficult because we have to memorize α in the case of $x \mathrel{+}= \alpha\{|\mathbf{p}_1 x_1, \epsilon|\}$. We should give another derivative to show the result like the corollary of PSPACE.)

This is all for my presentation.

bibliography I

Brzozowski, Janusz A (1964). "Derivatives of regular expressions". In: *Journal of the ACM (JACM)* 11.4, pp. 481–494.

Hoare, C.A.R.Tony et al. (2009). "Concurrent Kleene Algebra". English. In: *CONCUR 2009 - Concurrency Theory*. Ed. by Mario Bravetti and Gianluigi Zavattaro. Vol. 5710. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 399–414.

Hunt III, Harry B (1973). "On the time and tape complexity of languages I". In: *Proceedings of the fifth annual ACM symposium on Theory of computing*. ACM, pp. 10–19.

Jipsen, Peter (2014). "Concurrent Kleene algebra with tests". In: *Relational and Algebraic Methods in Computer Science*. Springer, pp. 37–48.

bibliography II

Kozen, Dexter and Frederick Smith (1996). "Kleene algebra with tests: Completeness and decidability". In: *Proc. 10th Int. Workshop Computer Science Logic (CSL'96)*. Ed. by D. van Dalen and M. Bezem. Vol. 1258. Lecture Notes in Computer Science. Utrecht, The Netherlands: Springer-Verlag, pp. 244–259.

Savitch, Walter J. (1970). "Relationships between nondeterministic and deterministic tape complexities". In: *Journal of Computer and System Sciences* 4.2, pp. 177–192.