Synchronous Kleene Algebra vs. Concurrent Kleene Algebra

Cristian Prisacariu

cristi@ifi.uio.no

Precise Modeling and Analysis group,
University of Oslo

Young Researchers Workshop on Concurrency Theory

5th of September 2009, Bologna, Italy.
Kleene Algebra
(Stephen C. Kleene, John H. Conway, Dexter Kozen, Ernie Cohen, ...)

- Kleene Algebra (KA) is the **equational theory of regular expressions**.
- KA is the structure \((A, +, \cdot, *, 0, 1)\) of actions with operations choice, sequence, and iteration.
- Models: regular sets, binary relations.
- Completeness [Kozen; Salomaa] of the axiomatization w.r.t. the models

KA is an idempotent semiring (under \(+, \cdot, 0, 1\)) satisfying axioms for *

\[
1 + \alpha \cdot \alpha^* \leq \alpha^* \quad \text{\(i.e., \alpha^* \beta\) is the least solution to \(\beta + \alpha \cdot X \leq X\)}
\]

\[
\beta + \alpha \cdot \gamma \leq \gamma \quad \rightarrow \quad \alpha^* \cdot \beta \leq \gamma
\]

where \(\alpha \leq \beta \triangleq \alpha + \beta = \beta\) is the natural order of an idem. semiring.

\[
\begin{align*}
\alpha + (\beta + \gamma) &= (\alpha + \beta) + \gamma \\
\alpha + \beta &= \beta + \alpha \\
\alpha + 0 &= 0 + \alpha = \alpha \\
\alpha + \alpha &= \alpha \\
\alpha \cdot (\beta \cdot \gamma) &= (\alpha \cdot \beta) \cdot \gamma \\
\alpha \cdot 1 &= 1 \cdot \alpha = \alpha \\
\alpha \cdot 0 &= 0 \cdot \alpha = 0 \\
\alpha \cdot (\beta + \gamma) &= \alpha \cdot \beta + \alpha \cdot \gamma \\
(\alpha + \beta) \cdot \gamma &= \alpha \cdot \gamma + \beta \cdot \gamma
\end{align*}
\]
Kleene Algebra

(Kleene Algebra (KA) is the equational theory of regular expressions.
KA is the structure \((A, +, \cdot, \ast, 0, 1)\) of actions with operations choice, sequence, and iteration.
Models: regular sets, binary relations.
Completeness [Kozen; Salomaa] of the axiomatization w.r.t. the models
KA is an idempotent semiring (under +, \cdot, 0, 1) satisfying axioms for *
\[1 + \alpha \cdot \alpha^* \leq \alpha^*\]
\[\beta + \alpha \cdot \gamma \leq \gamma \implies \alpha^* \cdot \beta \leq \gamma\]
where \(\alpha \leq \beta \triangleq \alpha + \beta = \beta\) is the natural order of an idem. semiring.

\[
\begin{align*}
\alpha + (\beta + \gamma) &= (\alpha + \beta) + \gamma \\
\alpha + \beta &= \beta + \alpha \\
\alpha + 0 &= 0 + \alpha = \alpha \\
\alpha + \alpha &= \alpha \\
\alpha \cdot (\beta \cdot \gamma) &= (\alpha \cdot \beta) \cdot \gamma \\
\alpha \cdot 1 &= 1 \cdot \alpha = \alpha \\
\alpha \cdot 0 &= 0 \cdot \alpha = 0 \\
\alpha \cdot (\beta + \gamma) &= \alpha \cdot \beta + \alpha \cdot \gamma \\
(\alpha + \beta) \cdot \gamma &= \alpha \cdot \gamma + \beta \cdot \gamma
\end{align*}
\]
Synchrony

(Robin Milner, Gerard Boudol, Gerard Berry, Georges Gonthier, ...)

- take the simple, clean model of synchrony from SCCS [Milner]
- instantaneous actions (events...); discrete time;
- determinism (from Esterel).

Esterel synchronous programming language [Berry et al.] grew up from SCCS, MEIJE, LUSTRE, ... and takes the synchrony model to extreme...
Now is the basis of the industrial framework SCADE.

\[
P \xrightarrow{a} P' \quad Q \xrightarrow{b} Q' \\
P \times Q \xrightarrow{a \times b} P' \times Q'
\]

Definition

In the synchronous model each and all of the concurrent systems execute instantaneously a single action at each time instant.
Motivation

Have an equational theory of “actions done at the same time” which has natural models on relational structures.

Why actions?

- A broad term: e.g.
  - instructions (as in programming languages),
  - human actions (as in legal contracts),
  - (as in distributed intelligent agents and communication protocols).

Why relational structures?

- Are the models of modal logics
  - logics of programs (like PDL, Hoare logics),
  - deontic logics,
  - logics for intelligent agents.

Our theory of synchronous actions can be the theoretical basis of all these logical formalisms over a simple concurrency model, the synchrony model.
You will see

1. **Concurrent Kleene Algebra (CKA)**
   - axiomatization of CKA
   - models of CKA as sets of traces (traces are sets of events)

2. **Synchronous Kleene Algebra (SKA)**
   - axiomatization of SKA
   - standard models of SKA as sets of synchronous strings

3. Discussions and Comparisons of the two formalisms
Concurrent Kleene Algebra

- **CKA** is a two quantales \((S, +, ;, 0, 1)\) and \((S, +, \ast, 0, 1)\) related by an exchange axiom.
  - **Quantale** = idempotent semiring that is complete lattice under \(\leq\)
  - In a quantale the Kleene \(\ast\) is defined with the least fixed point.
    \[
    \alpha^* \triangleq \mu X . 1 + \alpha; X
    \]
- ; - sequential composition \(\cdot\) in **SKA**
- \(\ast\) - concurrent composition \(\times\) in **SKA**

\[
(\alpha_1 \ast \beta_1); (\alpha_2 \ast \beta_2) \leq (\beta_1; \alpha_2) \ast (\alpha_1; \beta_2)
\]  (exchange axiom)

- Particular consequences of the exchange axiom:
  \[
  \alpha \ast \beta = \beta \ast \alpha \\
  (\alpha_1 \ast \beta_1); (\alpha_2 \ast \beta_2) \leq (\alpha_1; \alpha_2) \ast (\beta_1; \beta_2) \\
  \alpha; \beta \leq \alpha \ast \beta
  \]
Sets of sets of events as models of CKA

- Fix a set $E$ of events with a predefined dependence relation $\rightarrow$ (E.g.: $\rightarrow$ is data flow or control flow between events)
- A trace $t \subseteq E$ is a set of events.
- A program $P$ is a set of traces.
  - particular programs: $\text{skip} \triangleq \{\emptyset\}$, $e \triangleq \{\{e\}\}$, $\text{fail} \triangleq \emptyset$.
- Two operations on programs:
  - $P \ast Q = \{ tp \cup tq \mid tp \in P, tq \in Q, tp \cap tq = \emptyset \}$
  - $P ; Q = \{ tp \cup tq \mid tp \in P, tq \in Q, tp \cap tq = \emptyset, \text{dep}(tp) \cap tq = \emptyset \}$
- with properties:
  - $\ast$ is commutative. $;$ is not commutative
  - both $\ast$ and $;$ are monotone w.r.t. $\subseteq$
  - $\forall P, Q : P ; Q \subseteq P \ast Q$ concurrent behav. includes sequential behav.

$$(\mathcal{P}(\mathcal{P}(E)), \cup, ;, \ast, \text{fail, skip})$$ is a CKA
Synchronous Kleene Algebra (SKA) axiomatization

- $\alpha \in \mathcal{A}$ actions, $a \in \mathcal{A}_B$ (finite set of basic actions):
  $$\alpha ::= a \mid 0 \mid 1 \mid \alpha + \alpha \mid \alpha \cdot \alpha \mid \alpha \times \alpha \mid \alpha^*$$

- SKA is formed from two idempotent semirings $(\mathcal{A}, +, \cdot, 0, 1)$ respectively $(\mathcal{A}, +, \times, 0, 1)$

- respecting the Kleene $\ast$ axioms and two extra axioms for $\times$:
  $$\alpha \times \beta = \beta \times \alpha \quad \text{commutativity of } \times$$
  $$a \times a = a \quad \forall a \in \mathcal{A}_B \quad \text{restricted idempotence for } \times$$

- the two semirings are related by a synchrony axiom
  $$\left( \alpha_\times \cdot \alpha \right) \times \left( \beta_\times \cdot \beta \right) = \left( \alpha_\times \times \beta_\times \right) \cdot \left( \alpha \times \beta \right) \quad \forall \alpha_\times, \beta_\times \in \mathcal{A}_B^\times$$
Sets of synchronous strings as models

- A synchronous string over \( A_B \) is \( u, v \in (\mathcal{P}(A_B) \setminus \{\emptyset\})^* \) a string of non-empty sets of elements of \( A_B \) (notation \( x_i, y_j \in \mathcal{P}(A_B) \)).
- A synchronous set is a set of synchronous strings

Operations on synchronous sets:

\[
\begin{align*}
0 & \triangleq \emptyset & 1 & \triangleq \{\emptyset\} \quad \text{(skip)} & a & \triangleq \{\{a\}\} \quad \text{(basic action)} \\
A + B & \triangleq A \cup B & A \cdot B & \triangleq \{uv \mid u \in A, v \in B\} \quad \text{(string concatenation)} \\
A \times B & \triangleq \{u \times v \mid u \in A, v \in B\} \\
A^* & \triangleq \bigcup_{n \geq 0} A^n
\end{align*}
\]

\[
\begin{align*}
u \times \emptyset & \triangleq \emptyset \times u \triangleq u \\
u \times v & \triangleq (x \cup y)(u' \times v') \quad \text{where} \ u = xu' \ \text{and} \ v = yv'
\end{align*}
\]
**CKA vs. SKA**

Similarities:

- Extensions of Kleene algebra as **two idempotent semirings** where the concurrency/synchrony operation is **commutative**.
- Both can **encode Hoare-style reasoning** about sequential programs and about some form of concurrent programs.
CKA vs. SKA

Differences:

- CKA’s exchange axiom and SKA’s synchrony axiom cannot be related
- Exchange axiom is more general, applied to any actions
  Synchrony axiom is restricted to \( \times \)-actions (first elements of sequences)
- Exchange axiom is less informative (in terms of \( \leq \))
  Synchrony axiom expresses identities (in terms of \( = \)).

\[
(a_1 \ast b_1); (a_2 \ast b_2) \leq (b_1; a_2) \ast (a_1; b_2)
\]

\[
(a_x \cdot a) \times (b_x \cdot b) = (a_x \times b_x) \cdot (a \times b) \quad \forall a_x, b_x \in A_B^x
\]
CKA vs. SKA

Differences:

- CKA’s exchange axiom and SKA’s synchrony axiom cannot be related.
- Exchange axiom is more general, applied to any actions.
  Synchrony axiom is restricted to $\times$-actions (first elements of sequences).
- Exchange axiom is less informative (in terms of $\leq$).
  Synchrony axiom expresses identities (in terms of $=$).
- For CKA sequential behaviour is part of concurrent behaviour.
  For SKA sequential and synchronous behaviours are different.

$$\alpha; \beta \leq \alpha * \beta$$
**CKA vs. SKA**

Differences:

- **CKA’s exchange axiom** and **SKA’s synchrony axiom** cannot be related
- Exchange axiom is more general, **applied to any actions**
  Synchrony axiom is **restricted to \( \times \)-actions** (first elements of sequences)
- Exchange axiom is **less informative** (in terms of \( \leq \))
  Synchrony axiom **expresses identities** (in terms of \( = \)).
- For **CKA** sequential behaviour is part of concurrent behaviour
  For **SKA** sequential and synchronous behaviours are different
- In the models:
  - Traces vs. Synchronous strings (i.e., sets with a predefined arbitrary relation vs. sets with a restricted well defined structure)
  - Separation vs. Construction (i.e., in **CKA** the dependency relation is not changed vs. in **SKA** the relation defining the structure of the synchronous strings is constructed)
CKA and SKA vs. partial orders models

- CKA models have a very general dependency relation $\Rightarrow$ could not compare with the pomsets
- SKA synchronous strings are a class of pomsets

Synchronous strings are completely characterized by synchronous pomsets. In a synchronous pomset the partial order respects the restriction:

- all *maximal independent sets* are disjoint,
- uniquely labeled, and
- completely ordered

\[
\begin{array}{c}
\begin{array}{c}
e_i^1 \\
e_i^2 \\
\end{array}
& \quad \cdot \quad & \begin{array}{c}
e_j^1 \\
e_j^2 \\
\end{array}
\end{array}
\quad \cdot \\
\begin{array}{c}
X_i \\
\end{array}
\quad \cdot \\
\begin{array}{c}
X_j \\
\end{array}
\quad \cdot \quad \cdot \quad \cdot 
\]
You have seen

- **Kleene Algebra** combined with the notion of **Synchrony** (as in Milner’s SCCS or Esterel) we call it **Synchronous Kleene Algebra (SKA)**
  - axiomatization of **SKA**
  - standard models of **SKA** as sets of synchronous strings

- **Concurrent Kleene Algebra**
  - axiomatization with **quantales** and
  - general models as sets of sets of events with a general unrestricted dependence relation on the events.

- Discussions on the relations between the two formalisms.
  (not too formal)
Related Work

- not easy to integrate in logics of programs
  - Milner’s SCCS: axiomatization through quotient on a bisimulation;
  - French school: Esterel programming language and SCADE framework for synchronous programming;

- models of true concurrency
  - pomsets [Pratt]
  - Mazurkiewicz traces
  - event structures [Winskel et al.]

- other algebraic systems
  - mCRL2 [Groote et al.]
  - Q-algebras and constraint semirings

- logical formalisms with concurrency notions
  - PDL and CPDL [Peleg]
  - Dynamic Deontic Logic [Meyer]
  - Separation Logic
Thank you!