Algebraic Trace Theory

EE249
Presented by Roberto Passerone


Outline

- Introduction
- Concurrency Algebras
- Trace Algebras
- Trace Structure Algebras
- Conservative Approximations
- Homomorphisms
Abstract Algebra

- $x + 3 = 5$
- $(x + 3) + (-3) = 5 + (-3)$
- $x + (3 + (-3)) = 2$
- $x + 0 = 2$
- $x = 2$

Uniqueness of Solution

- "Addition" is associative
- There is an identity element
- There is an inverse for each element
- No need for commutativity!
- Axiomatic approach: don't define the objects, define their properties
- Many possible examples:
  - Integers under addition ($\mathbb{Z}$, $+$, 0, -)
  - Positive reals under multiplication ($\mathbb{R}^+$, $\times$, 1, $^{-1}$)
  - Permutations under functional composition ($F$, $\circ$, Id, $^{-1}$)
Algebra: abstractions

- Concrete: a carrier and the operations on it are given
- Abstract: a signature and the properties of the operations are given
- Universal: only the signature is given (studies algebraic structures in general)

Outline

- Introduction
- Concurrency Algebras
- Trace Algebras
- Trace Structure Algebras
- Conservative Approximations
- Homomorphisms
Objective

- Create models for agent behaviors
- Study the relationships between different models

A Concrete Model

- Define a domain of agents
  - The set of finite state automata
- Defines an interface for each agent
  - A set of signal names
- Defines a mechanism for instantiating agents
  - A signal renaming function
- Defines a mechanism for combining agents
  - e.g. product machine
- Defines a mechanism for encapsulating agents
  - Hide local signal names from the interface
An Abstract Algebraic Model

- Concurrency Algebra
  - The domain of agents is generic (carrier set)
  - Operation of parallel composition on agents
  - Operation of projection on agents
    - encapsulation, scoping
  - Operation of renaming on agents
    - instantiation
- Note: we don’t say what the carrier and the operations are, just that they must exist!
- But we must provide some properties that we want satisfied

Concurrency Algebra

- Let $W$ be a set of signal names
- A signature $\gamma$ is a pair $(I, O)$ where $I$ and $O$ are subsets of $W$
- We call the set $A = I \cup O$ the alphabet of a signature $\gamma$
- A renaming function $r : A \rightarrow B$ is a bijection between $A$ and $B$ (subsets of $W$)
Concurrency Algebra

- A concurrency algebra over W has a domain D of agents and the operations of
  - Parallel composition
  - Projection
  - Renaming
- With each agent in D is associated a signature γ
- Let E and E’ be agents, with signature γ = (I, O) and γ’ = (I’, O’)

Definition of the functions

- If O and O’ are disjoint, then E || E’ is defined and the signature is
  - (I ∪ I’, - (O ∪ O’), O ∪ O’)
- If I ⊆ B ⊆ A, then proj(B)(E) is defined and its signature is (I, O ∩ B)
- If r is a renaming function with domain A, then rename(r)(E) is defined and its signature is (r(I), r(O)) (where r is naturally extended to sets)
Axioms

C1. Associativity of composition
- \(( E || E') || E'' = E || ( E' || E'' )\)

C2. Commutativity of composition
- \(E || E' = E' || E\)

C3. Application of renaming functions
- \(\text{rename}( r )( \text{rename}( r' )( E ) ) = \text{rename}( r \circ r' )( E )\)

C4. Renaming and parallel composition commute
- \(\text{rename}( r )( E || E' ) = \text{rename}( r|_A )( E ) || \text{rename}( r|_{A'} )( E' )\)

C5. Identity of renaming
- \(\text{rename}( \text{id}_A )( E ) = E\)

C6. Application of projections
- \(\text{proj}( B )( \text{proj}( B' )( E ) ) = \text{proj}( B )( E )\)
Axioms

C7. Identity of projection

\[ \proj(A)(E) = E \]

C8. Projection and parallel composition commute

\[ \proj(B)(E || E') = \proj(B \cap A)(E) || \proj(B \cap A')(E') \]

if \( A \cap A' \subseteq B \)

C9. Projection and renaming commute

\[ \proj(r(B))(\rename(r)(E)) = \rename(r|_B)(\proj(B)(E)) \]

Outline

- Introduction
- Concurrency Algebras
- Trace Algebras
- Trace Structure Algebras
- Conservative Approximations
- Homomorphisms
Objective of Trace Theory

- Start from models of individual behaviors
- Construct models of agents that form a concurrency algebra
- Construct relationships between models of agents from relationships between individual behaviors

Trace and Trace Structure Algebras

- Trace algebra
  - Set of traces
  - Projection of traces
  - Renaming of traces

- Trace structure algebra
  - Model of individual behaviors
  - Model of agents

- Concurrency Algebra
  - Set of agents
  - Parallel composition of agents
  - Projection of agents
  - Renaming of agents
Question:

Q: In the Tagged Signal Model projection and composition are defined differently. How come?

A: The Tagged Signal Model does define projection and composition. We haven’t looked at a concrete example of an algebra yet!

Terminology

Injection: a one-to-one mapping
- If \( f( a ) = f( b ) \) then \( a = b \)

Surjection: a onto mapping
- For all elements \( a \) of the image of \( f \), there is \( x \) in the domain of \( f \) such that \( a = f( x ) \)

Bijection: a one-to-one and onto mapping
- Domain and image of \( f \) must be the same size
- Just like renaming the elements of the domain into the image using \( f \) as the correspondence
What Is a Trace?

- An element of a set of objects with the following two operations
  - projection
  - renaming
- Satisfying a set of axioms
- You must see a pattern here
  - Don’t define the objects, define their properties!
- A trace algebra is a mathematical structure (a set plus the operations) that satisfies the axioms

Trace Algebra

- A trace algebra $C_c$ over $W$ is a triple $(B_c, \text{proj}, \text{rename})$ where
  - For every $A \subseteq W$, $B_c(A)$ is the set of traces over $A$
  - Also use $B_c$ to denote all traces
  - $\text{proj}(B_c)$ and $\text{rename}(r)$ are partial functions from $B_c$ to $B_c$
Axioms of Trace Algebra

T1. Definition of projection
- \( \text{proj}(B)(x) \) is defined if there is \( A \) such that \( x \in B_C(A) \) and \( B \subseteq A \). When defined \( \text{proj}(B)(x) \) is an element of \( B_C(B) \).

T2. Application of projection
- \( \text{proj}(B)(\text{proj}(B')(x)) = \text{proj}(B)(x) \)

T3. Identity of projection
- If \( x \in B_C(A) \) then \( \text{proj}(A)(x) = x \)

Axioms of Trace Algebra

T5. Definition of renaming
- \( \text{rename}(r)(x) \) is defined if \( x \in B_C(\text{dom}(r)) \). When defined, \( \text{rename}(r)(x) \) is an element of \( B_C(\text{codom}(r)) \).

T6. Application of renaming
- \( \text{rename}(r)(\text{rename}(r')(x)) = \text{rename}(r \circ r')(x) \)

T7. Identity of renaming
- If \( x \in B_C(A) \) then \( \text{rename}(\text{id}_A)(x) = x \)
Axioms of Trace Algebra

T9. Projection and renaming commute

\[ \text{proj}\left( r \left( B \right) \right)\left( \text{rename}\left( r \right)\left( x \right) \right) = \text{rename}\left( r\big|_B \right)\left( \text{proj}\left( B \right)\left( x \right) \right) \]

T4. Diamond property

Let \( x \in B_C\left( A \right) \) and \( x' \in B_C\left( A' \right) \) be such that

\[ \text{proj}\left( A \cap A' \right)\left( x \right) = \text{proj}\left( A \cap A' \right)\left( x' \right). \]

For all \( A'' \supseteq A \cup A' \) there is \( x'' \in B_C\left( A'' \right) \) such that

\[ x = \text{proj}\left( A \right)\left( x'' \right) \text{ and } x' = \text{proj}\left( A' \right)\left( x'' \right) \]

Examples of trace algebras

\( C^l = \left( B^l_C, \text{proj}^l, \text{rename}^l \right) \)

For every \( A \), the set of traces \( B^l_C\left( A \right) \) of traces over \( A \) is \( A^\omega \) (finite and infinite sequences over \( A \))

\( \text{proj}^l\left( B \right)\left( x \right) \) is the sequence formed by removing every symbol \( a \) not in \( B \) (the sequence shrinks)

\( \text{rename}^l\left( r \right)\left( x \right) \) is the sequence formed by applying \( r \) to each element of the sequence
Examples of Trace Algebras

- The singleton \{ x_0 \}
  - Here all behaviors are the same. It conveys no information
- \( B_C( A ) = 2^A \)
  - \( \text{proj}( B )( x ) = x \cap B \)
  - \( \text{rename}( r )( x ) \) is the natural extension to sets
  - Abstracts time and retains only occurrence information

Examples of Trace Algebras

- \( B_C( A ) = ( 2^A )^\omega \)
  - \( \text{proj}( B )( x ) \) is intersection with \( B \)
  - Events carry a time stamp (Tagged Signal Model)
- \( B_C( A ) = \{ x( t ) \mid x : R^+ \rightarrow 2^A \} \)
  - Real-valued time stamps
  - Projection is the restriction of the image
Examples of Trace Algebras

- Interpret the set of signals $A$ as a set of state variables
  - A state is an assignment $\sigma : A \rightarrow V$ where $V$ is the set of possible values of the state variables
  - $B_C(A) = (A \rightarrow V)^\omega$
    - Sequence of assignment functions
    - Projection is a restriction of the domain of the assignment function

Outline

- Introduction
- Concurrency Algebras
- Trace Algebras
- Trace Structure Algebras
- Conservative Approximations
- Homomorphisms
Trace and Trace Structure Algebras

- Trace Algebra
  - Set of traces
  - Projection of traces
  - Renaming of traces

- Trace structure algebra
  - A trace structure contains a set of traces

- Concurrency Algebra
  - Set of agents
  - Parallel composition of agents
  - Projection of agents
  - Renaming of agents

Model of individual behaviors

Model of agents

Constructing agents

- We want to go from individual behaviors to models of agents
- We want the model of agents to be a concurrency algebra
  - Set of agents, with parallel composition, projection and renaming satisfying the axioms
- Naturally if a trace is an individual behavior, and agent is a set of those behaviors
Trace Structures

- A trace structure (or agent) over a trace algebra $C$ is the set of ordered pairs $(\gamma, P)$ where:
  - $\gamma$ is a signature over $W$ (reminder: $\gamma = (I, O)$)
  - $A$ is the alphabet of $\gamma$
  - $P$ is a subset of $B_C(A)$
- In other words, $P$ is a set of traces over the alphabet $A$
- Each trace in $P$ represents a possible behavior of the agent

**Note:** $||$, $\text{proj}^{TS}$ and $\text{rename}^{TS}$ denote operations on trace structures, which are derived from the corresponding operations on traces.

**A Trace Structure Algebra**

- $\mathcal{T}$ be a set of trace structures (agents)
- A Trace Structure Algebra is the 4-tuple $(\mathcal{T}, ||, \text{proj}^{TS}, \text{rename}^{TS})$ where
  - For $T'' = T || T'$: $\gamma'' = ((I \cup I') - (O \cup O'), O \cup O')$
  - $P'' = \{ x \in B_C(A'') : \text{proj}^{T}(A')(x) \in P$ and $\text{proj}^{T'}(A')(x) \in P' \}$
  - $\text{proj}^{TS}(B)(T) = (\text{proj}^{T}(B)(P))$
  - $\text{rename}^{TS}(r)(T) = (\text{rename}^{T}(r)(P))$
Example

\[ B_C( A ) = A^\infty \]

- \[ T = ( ( \{ a, b \}, \emptyset ), \{ abab \} ) \]
- \[ T' = ( ( \{ b, c \}, \emptyset ), \{ bcb \} ) \]

\[ T'' = T \parallel T' \]

\[ P'' = \{ x \in B_C( A'' ) : \text{proj}( A')( x ) \in P \text{ and proj}( A')( x ) \in P' \} \]

\[ P'' = \{ abacbc, abcabc \} \]

Note: \( P'' \) is maximal such that its projections correspond to the agents that are being composed.

Example

\[ B_C( A ) = ( 2^A )^\infty \]

- \[ T = ( ( \{ a, b \}, \emptyset ), \{ \langle a, b \rangle, \{ a \}, \{ b \} \} ) \]
- \[ T' = ( ( \{ b, c \}, \emptyset ), \{ \langle b \rangle, \{ c \}, \{ b \} \} ) \]

\[ T'' = T \parallel T' \]

\[ P'' = \{ \langle a, b \rangle, \{ a, c \}, \{ b \} \} \]

Note that under this trace algebra (with time stamps) we don’t get non-determinism due to interleaving semantics.
Example

B_C( A ) = ( 2^A )^\omega  \quad (\text{also } = N \rightarrow 2^A )

- T = ( ( \{ a, b \}, \emptyset ), \{ < \{ a \}, \{ a \}, \{ b \} > \} )
- T' = ( ( \{ a, b \}, \emptyset ), \{ < \{ b \}, \{ a \}, \{ b \} > \} )

T'' = T \parallel T'

Need P'' \subseteq B_C( A ) such that
  - \text{proj}( A )( P''_1 ) = \{ a \} \quad \text{and}
  - \text{proj}( A )( P''_1 ) = \{ b \}

Obviously this isn't possible when projection is defined as set intersection

In other words, T and T' are incompatible

Theorem

If C is a Trace Algebra, then the Trace Structure Algebra derived from C is a Concurrency Algebra

- I.e. the operations of parallel composition, projection and renaming satisfy the axioms of concurrency algebra

Note: the result is independent of the particular Trace Algebra (models of behaviors) that we start from
Outline

- Introduction
- Concurrency Algebras
- Trace Algebras
- Trace Structure Algebras
- Conservative Approximations
- Homomorphisms

Trace and Trace Structure Algebras

Goal: Study the problem of heterogeneous interaction
Formalize the concepts of abstraction and refinement
Conservative Approximations

- Refinement: within a model, an agent $T_{impl}$ refines a specification agent $T_{spec}$ if and only if $T_{impl} \subseteq T_{spec}$
- Let $\Psi_u$ and $\Psi_l$ be a pair of mappings from a set of agents $D$ to a set of agents $D'$
- By definition, a pair of functions $\Psi_u$ and $\Psi_l$ is a conservative approximation if and only if
  - if $\Psi_u( T_{impl} ) \subseteq \Psi_l( T_{spec} )$ then $T_{impl} \subseteq T_{spec}$
- Note: this is a definition, not a result!

Intuitively $\Psi_u( T_{impl} )$ is an upper bound of the implementation (has more behaviors)

Intuitively $\Psi_l( T_{spec} )$ is a lower bound of the specification (has less behaviors)

Hence we are being conservative

False negatives are possible. False positives are ruled out

The verification at the more abstract level should be easier
Constructing Approximations

- How do we build a conservative approximation?
- As before, we prefer to work at the level of the individual behavior
Let \( T_{\text{spec}} \) and \( T_{\text{impl}} \) be trace structures in \( A \). Then

\[
\text{if } \Psi_u(T_{\text{impl}}) \subseteq \Psi_u(T_{\text{spec}}) \text{ then } T_{\text{impl}} \subseteq T_{\text{spec}}
\]

Homomorphisms on Traces

- Let \( C \) and \( C' \) be two Trace Algebras
- A homomorphism \( h \) is a function from the traces in \( C (B_C) \) to the traces in \( C' (B_{C'}) \) such that \( h \) commutes with the operations of projection and renaming
  - \( h(\text{proj}(B)(x)) = \text{proj}'(B)(h(x)) \)
  - \( h(\text{rename}(r)(x)) = \text{rename}'(r)(h(x)) \)
- The homomorphism \( h \) is in general many-to-one
- Hence \( C' \) is in general more abstract than \( C \)
Building the Upper Bound

- Let $P$ be a set of traces, and consider the natural extension to sets $h(P)$ of $h$
- Clearly $P \subseteq h^{-1}(h(P))$
  - Because $h$ is many-to-one
  - This indeed is an upper bound!
  - Equality holds if $h$ is on-to-one
- Hence define
  - $\Psi_u(T) = (\gamma, h(P))$
Building the Upper Bound

\[ h^{-1}(h(P)) \]

Building the Lower Bound

- We want \( P \supseteq h^{-1}(\text{lb of } P) \)
- If \( x \) is not in \( P \), then \( h(x) \) should not be in the lower bound of \( P \)
- Hence define
  \[ \Psi_l(T) = h(P) - h(B_C(A) - P) \]
- There is a tighter lower bound
Building the Lower Bound

\[
h(P) - h(B_C(A) - P)
\]

\[
h^{-1}(h(P) - h(B_C(A) - P))
\]

\[
h(B_C(A) - P)
\]

October 2, 2003  ee249  51

Theorem

- The pair of functions just defined form a conservative approximation
  - if \( \Psi_u(T_{impl}) \subseteq \Psi_l(T_{spec}) \) then \( T_{impl} \subseteq T_{spec} \)

This is an example of a conservative approximation induced by a homomorphism

October 2, 2003  ee249  52
Examples of homomorphisms

- From $B_C(\mathcal{A}) = A^\infty$ to $B_C(\mathcal{A}) = 2^A$
  - $h(x) = \{ a \mid a$ appears in $x$ at least once $\}$
- Prove that this is a homomorphism
  - Must show it commutes with projection and renaming
- What do the upper and lower bounds look like?

Outline

- Introduction
- Concurrency Algebras
- Trace Algebras
- Trace Structure Algebras
- Conservative Approximations
- Homomorphisms
- Conservative Approximations: inverses
Approximations: Inverses

- Apply $\Psi_u$
- Apply $\Psi_l$
- Consider $T$ such that $\Psi_u(T) = \Psi_l(T) = T'$

Then $\Psi_{inv}(T') = T$
Approximations: Inverses

- Apply $\Psi_u$
- Apply $\Psi_l$
- Consider $T$ such that $\Psi_u(T) = \Psi_l(T) = T'$
- Then $\Psi_{inv}(T') = T$
- Can be used to embed high-level model in low level

Combining MoCs

Want to compose $T_1$ and $T_2$ from different trace structure algebras

- Construct a third, more detailed trace algebra, with homomorphisms to the other two
- Construct a third trace structure algebra
- Construct cons. approximations and their inverses
- Map $T_1$ and $T_2$ to $T_1'$ and $T_2'$ in the third trace structure algebra
- Compose $T_1'$ and $T_2'$
Combining MoCs

Want to compose $T_1$ and $T_2$ from different trace structure algebras

- Compose $T_1'$ and $T_2'$
- Get $T''$
- Project back to $T_1'$ and $T_2'$
- Map back in the abstract domain

- Find restrictions due to the interaction at the higher level (constraint application)
- Find greatest possible $T_1$ and $T_2$ that still have the same interaction (don’t cares, synthesis)

Key Insight

- Using this process you can
  - Find restrictions due to the interaction at the higher level (constraint application)
  - Find greatest possible $T_1$ and $T_2$ that still have the same interaction (don’t cares, synthesis)
A Sample of Trace Algebras

\[ A^* \]
\[ (\varphi(A) - \{\emptyset\})^* \]
\[ \text{pomsets}(A) \]
\[ (A \rightarrow V)^* \quad \text{(sf: stutter free)} \]
\[ (A \rightarrow V)^* \]
\[ (\varphi(A))^* \]
\[ \varepsilon \rightarrow ((A_V \rightarrow V) \times \varphi(A_E)) \]
\[ \text{tomsets}((A_V \rightarrow V) \times \varphi(A_E)) \quad \text{(sf)} \]
\[ (A \rightarrow V)^2 \]
\[ \gamma \rightarrow (A \rightarrow V) \]
\[ \gamma \rightarrow \varphi(A) \]
\[ \gamma \rightarrow (A^*) \]
\[ A \rightarrow \{0, 1, X\} \]
\[ ((A_V \rightarrow V) \times (\varphi(A_V) - \{\emptyset\}))^* \quad \text{(sf)} \]

Interleaving semantics
Explicit simultaneity
Partial order models
State based asynchronous time
State based synchronous time
Event based synchronous time
Metric time
Non-metric time
Pre-post
State based discrete time
Event based discrete time
Discrete time with interleaving
Combinational
GALS

October 2, 2003

ee249

61