Propositional Stabilisation Theory

Interface Types for Causality and Timing Analyses

Michael Mendler
Informatics Theory Group
Information Systems and Applied Computer Sciences
The Otto-Friedrich-University of Bamberg
What is this talk about?

Special purpose type theory (PST) for component interfaces

- to express different forms of causal response behaviour
- resulting in different degrees of constructivity
- specifying various forms of data-dependent schedulability and timing analyses.

PST is

- purely propositional (enriching Boolean and Ternary Alg.)
- combining Time + Causality + Function
- of intuitionistic, 2nd-order expressiveness.
Constructiveness Analysis -- Pain-in-the-Neck, or Food-for-Thought?
Motivation

For all inputs there is a **unique stationary Boolean solution.** Thus, the system is **logically reactive.** However, the system is **not constructive.**
Motivation

\[
\begin{align*}
 s_1 &= \bot + x \\
 s_2 &= \bot + \bot \\
 s_3 &= \overline{x} \cdot \bot
\end{align*}
\]

For all inputs there is a unique stationary Boolean solution. Thus, the system is logically reactive. However, the system is not constructive.
Motivation

For all inputs there is a unique stationary Boolean solution. Thus, the system is logically reactive. However, the system is not constructive.
Motivation

For all inputs there is a unique stationary Boolean solution. Thus, the system is logically reactive. However, the system is not constructive.

But what if we are compiling for a component-based and distributed architecture?
Motivation

\[ s_1 = s_2 + x \]
\[ s_2 = \overline{s_1} + s_3 \]
\[ s_3 = \overline{x} \cdot s_2 \]

Oscillation under up-bounded inertial delay scheduling [Brzozowski & Seger]

Malta, 21-25 Nov. 2005
Constructiveness Analysis

The distributed, multi-threaded execution of a logically reactive P may produce anomalous behaviour:
  -- deadlocks, oscillation,
  -- non-determinism, metastability.

The problem may (often) be fixed at two levels:

Constrain Run-time System: Find a restricted schedule which avoids anomalies and guarantees stabilisation.
Constrain Code Generator: Harden P's code so it becomes constructive under arbitrary run-time schedules.
Example

\[ s_1 = s_2 + x \]
\[ s_2 = \overline{s_1} + s_3 \]
\[ s_3 = \overline{x} \cdot s_2 \]

Oscillation can be avoided if we
- schedule \( s_1, s_3 \) with higher priority than \( s_2 \) or
- implement \( s_1, s_3 \) atomically, as 2in/2out complex-gate.

Then, whenever \( s_2 \) is executed, we maintain the invariant
\[ s_2 = \overline{s_1} + s_3 = \overline{x} \]
Oscillation can be avoided if we
- schedule $s_1$, $s_3$ with higher priority than $s_2$ or
- implement $s_1$, $s_3$ atomically, as 2in/2out complex-gate.

Then, whenever $s_2$ is executed, we maintain the invariant
$$s_2 = \overline{s_1} + s_3 = \overline{x}$$

Alternatively, we may harden the code.
There are many “causality improving” transformations:

- e.g., Boussinot, Schneider:

\[ s \cdot f(s) \sqsubseteq s \cdot f(1) \]
\[ s \cdot f + \overline{s} \cdot g \sqsubseteq s \cdot f + \overline{s} \cdot g + f \cdot g \]

... and there should be more.

Now,

- A Theory of Causal Interface Types
- Semantical characterisation of degrees of causality
- Compositional analyses
Introducing PST Type Theory
Types

- intuitionistic modal logic (modal operator "\(\Diamond\)"")
- \(\Diamond M \text{ "true"} = M \text{ "valid in bounded time"}\)
Types

\[ M ::= a = v \mid M \wedge N \mid M \vee N \mid M \supset N \mid \neg M \mid o M \]

\[ a \in \text{Sig} \quad v \in \mathbb{B} \]
Specifying Reactions

\[ \text{KSystem} \subseteq \text{Sig} \rightarrow \mathbb{N} \rightarrow \mathbb{B} \]

\[ \text{KSystem} \models M \text{ iff } \exists \delta \in [M]. \forall V \in \text{KSystem}. V \models \delta : M \]

Semantics

- \( M \) stabilisation type (causality + function)
- \( \delta \in [M] \) timing constraint (\( \lambda \)-terms)
- \( V \models \delta : M \) waveform \( V \in \text{Sig} \rightarrow \mathbb{N} \rightarrow \mathbb{B} \) satisfies \( M \) with timing constraint \( \delta \in [M] \)
PST Timing Information

Type M

\(M \land N\) conjunction

\(M \lor N\) disjunction

\(M \supset N\) implication

\(\ominus M\) modality

\(a=v\) atomic

Timing Information \([M]\)

\([M \land N] = [M] \times [N]\)

cartesian product

\([M \lor N] = [M] + [N]\)

disjoint union

\([M \supset N] = [M] \rightarrow [N]\)

function space

\([\ominus M] = \mathbb{N} \times [M]\)

propagation delay

\([a=v] = 1\)

no information

Propositions-as-Types Principle
**PST Waveform Specification**

\[ V(a) \downarrow_t v \]

"Signal a stabilises to value v in waveform V as from time t"

\[ V^\delta \] is the time-shifted waveform \[ V^\delta(a)(t) = V(a)(t + \delta) \]

\[
\begin{align*}
V \models 0 : a=v & \iff V(a) \downarrow_0 v \\
V \models (c, d) : M \land N & \iff V \models c : M \text{ and } V \models d : N \\
V \models f & \in [M]. \\
V \models (\delta, c) : \circ M & \iff V^\delta \models c : M \\
V^\circ \models f(c) : N
\end{align*}
\]

Variation of the standard Realisability Interpretation for Intuitionistic Logic
In how many ways can we say an output responds with a Boolean?

**Predicate logic**

\[ \forall V. \exists v. \exists t. V(a) \downarrow_t v \]

\[ \exists v. \forall V. \exists t. V(a) \downarrow_t v \]

\[ \exists t. \forall V. \exists v. V(a) \downarrow_t v \]

\[ \exists v. \exists t. \forall V. V(a) \downarrow_t v \]

\[ \forall V. \exists v. \forall t. V(a) \downarrow_t v \]

\[ \exists v. \forall V. \forall t. V(a) \downarrow_t v \]

\[ \phi \oplus \psi \overset{\text{df}}{=} (\phi \supset \psi) \supset \psi \land ((\psi \supset \phi) \supset \phi) \]

**PST stabilisation type**

\[ \neg\neg(a=1 \oplus a=0) \]

\[ \neg\neg a=1 \lor \neg\neg a=0 \]

\[ \circ(a=1 \oplus a=0) \]

\[ \circ a=1 \lor \circ a=0 \]

\[ a=1 \oplus a=0 \]

\[ a=1 \lor a=0 \]
Three Levels of Signal Evaluation

\[ V^\infty(a) =_{df} \begin{cases} 
1 & \exists t. V(a) \downarrow_t 1 \\
0 & \exists t. V(a) \downarrow_t 0 \\
\bot & \text{otherwise}
\end{cases} \]

\[ f : \mathbb{B}^2 \to \mathbb{B} \quad \text{Boolean function} \]
\[ f^\infty : \mathbb{K}^2 \to \mathbb{K} \quad \text{ternary extension of} \quad f \]

### predicate logic

\[ \forall V. \exists t. \downarrow_t f^\infty(V^\infty(a), V^\infty(b)) \quad \neg\neg(c = f(a, b)) \]
\[ \exists t. \forall V. \downarrow_t f^\infty(V^\infty(a), V^\infty(b)) \quad \circ(c = f(a, b)) \]
\[ \forall t. \forall V. \downarrow_t f^\infty(V^\infty(a), V^\infty(b)) \quad c = f(a, b) \]
In how many "causal ways" can we produce a unique stationary response (logical correctness)?
In how many “causal ways” can we produce a unique stationary response (logical correctness)?

\[ M_1 = \circ(a=1 \land b=1 \land c=0) \land (a=1 \supset b=1) \land (b=1 \supset c=0) \land (c=0 \supset a=1) \]
In how many “causal ways” can we produce a unique stationary response (logical correctness)?

\[ \delta : M_{2} \]

\[ \text{max}(t_{a}, t_{b}) \leq t_{c} \leq \text{max}(t_{a}, t_{b}) + \delta \]

\[ M_{2} = ((a=1 \land b=1) \supset c=0) \land (c=0 \supset (a=1 \land b=1)) \land \neg\neg c=0 \]
In how many “causal ways” can we produce a unique stationary response (logical correctness)?

\[ M_3 = (c=0 \supset (\circ a=1 \land \circ b=1)) \land ((a=1 \lor b=1) \supset c=0) \land \neg \neg c=0 \]
Interfaces for Causality and Timing
Example — Stabilisation Models

d:AND data-independent "topological" model

\[
\begin{align*}
a & \downarrow \\
b & \Rightarrow \\
c & \\
d & \ Wealth \\
\end{align*}
\]

d timing information

d \in \text{Nat} \approx [ \text{AND} ]

AND type specification

\[(a = 1 \oplus a = 0) \land (b = 1 \oplus b = 0) \supset (c = 1 \oplus c = 0)\]
Example — Stabilisation Models

\[ d \text{: AND, data-dependent "topological" model} \]

\[
\begin{align*}
& \quad (a=1 \lor a=0) \land (b=1 \lor b=0) \supset \circ (c = a \cdot b)
\end{align*}
\]

\[ d \text{ timing information} \]

\[
d = (d_{00}, d_{01}, d_{10}, d_{11}) \in \text{Nat}^4 \approx [\text{AND}]\]
Example — Stabilisation Models

\[ d : \text{AND} \quad \text{data-dependent "floating mode" model} \]

\[ d_{1} \quad d_{2} \quad d_{3} \]

\[ a \quad b \quad c \]

Timing information:
\[ d = (d_{1}, d_{2}, d_{3}) \in \text{Nat}^{3} \approx [\text{AND}] \]

Type specification:
\[ ((a=1 \land b=1) \lor a=0 \lor b=0) \supset o(c = a \cdot b) \]
Example — Stabilisation Models

\[ d: \text{AND} \quad \text{data-independent "floating mode" model} \]

\[ \begin{array}{c}
\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c}
\end{array}
\end{array} \quad \begin{array}{c}
\begin{array}{c}
\text{1} \\
\text{1} \\
\text{1}
\end{array}
\end{array} \quad \begin{array}{c}
\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c}
\end{array}
\end{array} \quad \begin{array}{c}
\begin{array}{c}
\text{0} \\
\text{0} \\
\text{0}
\end{array}
\end{array} \quad \begin{array}{c}
\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c}
\end{array}
\end{array} \quad \begin{array}{c}
\begin{array}{c}
\text{0} \\
\text{0} \\
\text{0}
\end{array}
\end{array} \]

\[ d \text{ timing information} \]

\[ d \in \text{Nat} \approx [\text{AND}] \]

\[ \text{AND type specification} \]

\[ ((a=1 \land b=1) \oplus a=0 \oplus b=0) \supset o(c = a \cdot b) \]
Example — Stabilisation Models

**Example — Stabilisation Models**

- **d**: \( \text{AND} \) data-dependent "static sensitization" model

  \[ d = (d_1, d_2) \in \text{Nat}^2 \approx [\text{AND}] \]

- **Timing information**

  \[ d \text{ timing information} \]

- **AND type specification**

  \[ ((\neg \neg a = 1 \land b = v) \lor (a = v \land \neg \neg b = 1)) \supset \circ (c = v) \]
Summary PST

- PST types are intuitionistic, fully compatible with ternary model.
- The difference between reactive and stationary behaviour is the difference between $M$ and $\neg \neg M$.
- A fixed stationary behaviour $\neg \neg M$ can be implemented by various reactive types (causal/timing models) $M_1$, $M_2$, $M_3$ ..., such that
  $$\neg \neg M_1 \equiv \neg \neg M, \quad \neg \neg M_2 \equiv \neg \neg M, \quad \neg \neg M_3 \equiv \neg \neg M, \ldots$$
- PST can characterise different timing analyses...
PST Timing Analyses
PST Timing Analysis

semantical or syntactical deduction in PST type synthesis and type transformation
Conclusion
PST Reactiveness Analysis — Advantages

- **Adjustable granularity of data abstraction**
- **Compositionality** (= "divide and conquer" analyses)
- **Precision**
  - semantical meaning of computed delays is specified uniquely (= data type, type checking)
- **"Lossless" heuristics**
  - free exploration of search space through combination of partial (i.e., incorrect or suboptimal) analyses
Deduction in PST captures correct and exact timing analyses for all *elementary* combinational stabilisation models.

In this fragment a number of well-known analyses can be characterised:

- Topological
- Statical [Benkoski et.al. '90]
- Polynomial [Huang et.al. '91]
- Floating [Chen&Du '90, Devadas et.al. '91]
- Viability [McGeer '89]
Open Problems — Projects, PhD Topics

• **Theory**
  
  Complete characterisation of PST expressiveness
  Axiomatization

• **Implementation**
  
  Efficient data structures for fragments of PST
  Toolbox of composable (partial) heuristics

• **Application**
  
  Explore links: PST analyses – degrees of causality – distributed code generation

