Kleene Algebra — Lecture 2
Last lecture

- We discussed rational expressions and their semantics.
- We introduced a set of sound laws, and used those in proofs.
- We studied the relational and language models.
Today’s lecture

- Focus on “filter programs” — the ones that “do nothing or crash”.
- They admit their own equations, useful for reasoning about programs.
- Extend syntax and reasoning to obtain *Kleene Algebra with Tests*. 
Filter programs

Suppose \( a, b \in \Sigma \) are “filtering” programs, i.e.:

\[
\sigma(a) = \{ \langle s, s \rangle \in S \times S : \phi(s) \} \quad \text{and} \quad \sigma(b) = \{ \langle s, s \rangle \in S \times S : \psi(s) \}.
\]

In that case, you can show

\[
[a \cdot b]_{\sigma} = \{ \langle s, s \rangle \in S \times S : \phi(s) \land \psi(s) \} = [b \cdot a]_{\sigma}.
\]

Filtering programs admit more of these useful specialized equalities.
Extended syntax

We fix a set $T = \{t, s, \ldots\}$ of primitive tests.

**Definition (Boolean expressions)**
We write $\mathbb{B}$ for the set of *Boolean expressions*, generated by

$$\mathbb{B} \ni b, c ::= 0 \mid 1 \mid t \in T \mid b + c \mid b \cdot c \mid \overline{b}$$

**Definition (Guarded rational expressions)**
We write $\mathbb{G}$ for the set of *guarded rational expressions*, generated by

$$\mathbb{G} \ni e, f ::= b \in \mathbb{B} \mid a \in \Sigma \mid e + f \mid e \cdot f \mid e^*$$
Definition (Guarded interpretation)

A *(guarded) interpretation* is a triple \( \langle S, \tau, \sigma \rangle \), where

- \( \langle S, \sigma \rangle \) is an interpretation; and
- \( \tau : T \rightarrow 2^S \) is a function.

Intuitively: \( \tau(t) \) is the set of states where \( t \) holds.
Definition (Boolean expression semantics)
Let $\langle S, \tau, \sigma \rangle$ be a guarded interpretation. We define $(\cdot)_\sigma : \mathbb{B} \rightarrow 2^S$ inductively:

\[
\begin{align*}
(0)_\tau &= \emptyset \\
(1)_\tau &= S \\
(t)_\tau &= \tau(t) \\
(b + c)_\tau &= (b)_\tau \cup (c)_\tau \\
(b \cdot c)_\tau &= (b)_\tau \cap (c)_\tau \\
(\overline{b})_\tau &= S \setminus (b)_\tau
\end{align*}
\]
Definition (Guarded rational expression semantics)
Let $\langle S, \tau, \sigma \rangle$ be a guarded interpretation. We define $\llbracket - \rrbracket_{\sigma, \tau} : G \to 2^{S \times S}$ inductively:

\[
\begin{align*}
\llbracket b \rrbracket_{\sigma, \tau} &= \{ \langle s, s \rangle : s \in \llbracket b \rrbracket_{\tau} \} \\
\llbracket a \rrbracket_{\sigma, \tau} &= \sigma(\! a \!)
\end{align*}
\]

\[
\begin{align*}
\llbracket e + f \rrbracket_{\sigma, \tau} &= \llbracket e \rrbracket_{\sigma, \tau} \cup \llbracket f \rrbracket_{\sigma, \tau} \\
\llbracket e \cdot f \rrbracket_{\sigma, \tau} &= \llbracket e \rrbracket_{\sigma, \tau} \circ \llbracket f \rrbracket_{\sigma, \tau} \\
\llbracket e^* \rrbracket_{\sigma, \tau} &= \llbracket e \rrbracket^*_{\sigma, \tau}
\end{align*}
\]
Consider our integer square root finding program from the last lecture:

\[
i \leftarrow 0; \\
\text{while } (i + 1)^2 \leq n \text{ do} \\
\quad i \leftarrow i + 1;
\]

To encode this program, we had to encode both the loop guard and its negation:

\[
\sigma(\text{guard}) = \{ \langle s, s \rangle : (s(i) + 1)^2 \leq s(n) \} \\
\sigma(\text{validate}) = \{ \langle s, s \rangle : (s(i) + 1)^2 > s(n) \}
\]
New, shiny encoding:

\[ \text{init} \cdot (\text{bound} \cdot \text{incr})^* \cdot \text{bound} \]

Let \( T = \{\text{bound}\} \) and \( \Sigma = \{\text{init}, \text{incr}\} \), and set

\[ \tau(\text{bound}) = \{s \in S : (s(i) + 1)^2 \leq s(n)\} \]

\[ \sigma(\text{init}) = \{\langle s, s[0/i]\rangle : s \in S\} \]

\[ \sigma(\text{incr}) = \{\langle s, s[s(i) + 1/i]\rangle : s \in S\} \]
We can encode traditional flow control:

\[
\begin{align*}
\text{if } b \text{ then } e \text{ else } f & := b \cdot e + \overline{b} \cdot f \\
\text{if } b \text{ then } e & := b \cdot e + \overline{b} \\
\text{while } b \text{ do } e & := (b \cdot e)^* \cdot \overline{b}
\end{align*}
\]

This means our program can be written as

\[
\text{init} \cdot \textbf{while} \ \text{bound} \ \textbf{do} \ \text{incr}
\]
Boolean equivalence — laws

Definition (Boolean algebra)

$\equiv_B$ is the smallest congruence on $\mathbb{B}$ satisfying, for all $b, c, d \in \mathbb{B}$:

\[
\begin{align*}
    b + 0 & \equiv_B b & b + c & \equiv_B c + b & b + \overline{b} & \equiv_B 1 & b + (c + d) & \equiv_B (b + c) + d \\
    b \cdot 1 & \equiv_B b & b \cdot c & \equiv_B c \cdot b & b \cdot \overline{b} & \equiv_B 0 & b \cdot (c \cdot d) & \equiv_B (b \cdot c) \cdot d \\
    b + c \cdot d & \equiv_B (b + c) \cdot (b + d) & b \cdot (c + d) & \equiv_B b \cdot c + b \cdot d
\end{align*}
\]
Boolean equivalence — soundness

Lemma (Soundness for Boolean algebra)

Let \( b, c \in \mathbb{B} \), and let \( \tau : T \to 2^S \). If \( b \equiv_b c \), then \( (b)_\tau = (c)_\tau \).
Lemma (Opposites)

If $b, c \in \mathbb{B}$ such that $b + c \equiv_{\mathbb{B}} 1$ and $b \cdot c \equiv_{\mathbb{B}} 0$, then $b \equiv_{\mathbb{B}} \overline{c}$. 
Lemma (DeMorgan’s first law)

If $b, c \in \mathbb{B}$, then $b + c \equiv \overline{b} \cdot \overline{c}$. 
Guarded rational equivalence — laws

Definition (Kleene Algebra with Tests)

≡_G sas the smallest congruence on G satisfying, for all b, c ∈ B and e, f, g ∈ G:

\[ b \equiv_B c \implies b \equiv_G c \quad e + 0 \equiv_G e \quad e + e \equiv_G e \quad e + f \equiv_G f + e \]
\[ e + (f + g) \equiv_G (e + f) + g \quad e \cdot (f \cdot g) \equiv_G (e \cdot f) \cdot g \]
\[ e \cdot (f + g) \equiv_G e \cdot f + e \cdot g \quad (e + f) \cdot g \equiv_G e \cdot g + f \cdot g \]
\[ e \cdot 1 \equiv_G e \equiv_G 1 \cdot e \quad e \cdot 0 \equiv_G 0 \equiv_G 0 \cdot e \quad 1 + e \cdot e^* \equiv_G e^* \equiv_G 1 + e^* \cdot e \]
\[ e + f \cdot g \leq_G g \implies f^* \cdot e \leq_G g \quad e + f \cdot g \leq_G f \implies e \cdot g^* \leq_G f \]
Guar ded rational equivalence — soundness

Lemma (Soundness for Kleene Algebra with Tests)
Let $e, f \in \mathcal{G}$, and let $\langle S, \tau, \sigma \rangle$ be a guarded interpretation.

If $e \equiv_G f$, then $[e]_{\sigma,\tau} = [f]_{\sigma,\tau}$. 
Lemma (Branch swapping)

Let \( e, f \in \mathbb{G} \) as well as \( b \in \mathbb{B} \). The following holds:

\[
\text{if } b \text{ then } e \text{ else } f \equiv \text{if } \overline{b} \text{ then } f \text{ else } e
\]

Proof.
This is a matter of unrolling the syntactic sugar and applying our rules:

\[
\begin{align*}
\text{if } b \text{ then } e \text{ else } f &= b \cdot e + \overline{b} \cdot f \\
&\equiv \overline{b} \cdot f + b \cdot e \quad \text{(by definition)} \\
&\equiv \overline{b} \cdot f + \overline{\overline{b}} \cdot e \quad \text{(commutativity)} \\
&\equiv \overline{b} \cdot f + \overline{b} \cdot e \quad \text{(see homework)} \\
&= \text{if } \overline{b} \text{ then } f \text{ else } e \quad \text{(by definition)}
\end{align*}
\]
Lemma (Loop unrolling)

Let $e \in G$ and $b \in B$. The following holds:

$$\text{while } b \text{ do } e \equiv \text{if } b \text{ then } (e \cdot \text{while } b \text{ do } e)$$

Proof.

We again unfold the syntactic sugar and apply our rules, as follows:

$$\text{while } b \text{ do } e = (b \cdot e)^* \cdot \overline{b}$$

(by definition)

$$\equiv (b \cdot e \cdot (b \cdot e)^* + 1) \cdot \overline{b}$$

(unrolling)

$$\equiv b \cdot e \cdot (b \cdot e)^* \cdot \overline{b} + \overline{b}$$

(distributivity)

$$= \text{if } b \text{ then } (e \cdot \text{while } b \text{ do } e)$$

(by definition) □
Guarded rational equivalence — reasoning

You now have the tools to prove the equivalence claimed earlier:

\[
\begin{align*}
\text{while } a \text{ and } b \text{ do} & \\
& \quad e; \\
\text{while } a \text{ do} & \\
& \quad f; \\
& \quad \text{while } a \text{ and } b \text{ do} \\
& \quad e; \\
\end{align*}
\]
\[
\begin{align*}
\equiv \\
\text{while } a \text{ do} & \\
& \quad \text{if } b \text{ then} \\
& \quad \quad e; \\
& \quad \text{else} \\
& \quad \quad f;
\end{align*}
\]

This is part of today’s exercises.
We can abstract from guarded interpretations using guarded languages.

Idea: record the order of actions, and the tests that hold in-between.

**Definition (Guarded languages)**

A guarded word is a word over \((2^T \cup \Sigma)^*\) of the form

\[ \alpha_1 a_1 \alpha_2 a_2 \alpha_3 \cdots \alpha_{n-1} a_{n-1} \alpha_n \]

A guarded language is a set of guarded words.

We write \((\Sigma, T)^*\) for the set of guarded languages.
Guarded language semantics

Definition (Guarded language composition)
Let $L, K \subseteq (\Sigma, T)^*$. We write $L \diamond K$ for the guarded product:

$$L \diamond K = \{w \alpha x : \alpha \in 2^T, w \alpha \in L, \alpha x \in K\}$$

We write $L^{\diamond}$ for the guarded star:

$$L^{\diamond} = 2^T \cup L \cup L \diamond L \cup L \diamond L \diamond L \cup \cdots$$
Guarded language semantics

Definition (Guarded language semantics)
We define \( (-)_G : B \to 2^{2^T} \) inductively, as follows:

\[
\begin{align*}
(0)_G &= \emptyset \\
(1)_G &= 2^T \\
(t)_G &= \{ \alpha \in 2^T : t \in \alpha \} \\
(b + c)_G &= (b)_G \cup (c)_G \\
(b \cdot c)_G &= (b)_G \cap (c)_G \\
(\overline{b})_G &= S \setminus (b)_G
\end{align*}
\]

Next, we define \( [-]_G : G \to (\Sigma, T)^* \) inductively, as follows:

\[
\begin{align*}
[b]_G &= (b)_G \\
[a]_G &= \{ a \} \\
[e + f]_G &= [e]_G \cup [f]_G \\
[e \cdot f]_G &= [e]_G \diamond [f]_G \\
[e^*]_G &= [e]_G^{(\diamond)}
\end{align*}
\]
Theorem (Equivalence of guarded models)

Let $e, f \in \mathbb{G}$. The following are equivalent:

(i) $\llbracket e \rrbracket_\mathbb{G} = \llbracket f \rrbracket_\mathbb{G}$

(ii) for all $\sigma$ and $\tau$, $\llbracket e \rrbracket_{\sigma,\tau} = \llbracket f \rrbracket_{\sigma,\tau}$.

Proof.

Like in the last lecture, but with more Greek letters!
From now on

- No more guarded expressions!
- Everything still works when you add tests.
- The proofs just become more involved.
Next lecture

- Representing languages using *automata*.
- Checking language equivalence of automata.
- Converting expressions to automata.