Kleene Algebra — Lecture 3

Tobias Kappé

January 2022

1 Today's lecture

Over the previous two lectures, we've spent a lot of time and effort to prove certain equivalences. But what about *disproving* them? In other words, how can we show that there cannot possibly be a proof of a certain assertion, based on the rules of Kleene Algebra or Kleene Algebra with Tests? In this lecture, we will see one way of achieving this: construct a model that adheres to all of the laws of Kleene Algebra (with Tests), but not to the claimed equivalence.

However, constructing this model is not always easy, because we are left to guess at its semantic domain and interpretation. This is why we will introduce another model, that of *languages*, and show how it can be used to efficiently construct counterexamples to claimed equivalences. We will show that this language model accurately reflects all equalities that hold for relational models.

We finish this lectures with some thoughts about the limitations of this approach, and at least one hint of the material that will occupy us next week.

2 Refutation

Suppose you're working through a proof, and you find yourself in a situation where you have to verify some equivalence $e \equiv f$ in order to proceed. You give it your level best, but try as you might, the proof just does not go through. Could it be that you've taken a wrong turn, and what you're trying to establish is actually false? Before abandoning your proof, you decide you want to be sure that you're not just missing something: can you show that $e \not\equiv f$?

Proving a negative is not always easy; as far as $\equiv$ goes, you only have rules that establish equivalence, not inequivalence. The only real way forward is to use a proof by contradiction: assume that $e \equiv f$ *does* hold, and somehow derive something that is patently false. If you go this way, the soundness lemmas that we have seen in previous lectures can be very useful. Let's review them.

Lemma 3.1 (Soundness overview)**.** *Let $\langle S, \sigma, \tau \rangle$ be a guarded interpretation (note that this makes $\langle S, \sigma \rangle$ an interpretation). The following hold:*

1. *For all $b, c \in \mathbb{B}$, if $b \equiv f$, then $(\!|e|\!)_\tau = (\!|f|\!)_\tau$.*
2. *For all $e, f \in \mathbb{E}$, if $e \equiv f$, then $[\![e]\!]_\sigma = [\![f]\!]_\sigma$.*
3. *For all $e, f \in \mathbb{G}$, if $e \equiv f$, then $[\![e]\!]_{\sigma,\tau} = [\![f]\!]_{\sigma,\tau}$.*

So, assuming that $e \equiv f$, the above tells us that e and f have the same semantics under any interpretation. If you want to reach a contradiction, all you have to do is construct an interpretation for which this is not true.

Let's see an example of using soundness for refutation. Let $\mathtt{t}, \mathtt{s} \in T$, and suppose we want to know whether $\overline{\mathtt{t}} \cdot \mathtt{s} \equiv \overline{\mathtt{t} \cdot \overline{\mathtt{s}}}$. Let's choose $S = \{0,1\}^2$ and

$$\tau(\mathtt{t}) = \{\langle 1,0\rangle, \langle 1,1\rangle\} \qquad\qquad \tau(\mathtt{s}) = \{\langle 0,1\rangle, \langle 1,1\rangle\}$$

Then, if $\overline{\mathtt{t}} \cdot \mathtt{s} \equiv \overline{\mathtt{t} \cdot \overline{\mathtt{s}}}$, it should follow that $(\!|\overline{\mathtt{t}} \cdot \mathtt{s}|\!)_\tau = (\!|\overline{\mathtt{t} \cdot \overline{\mathtt{s}}}|\!)_\tau$. However, if we calculate, it turns out that $(\!|\overline{\mathtt{t}} \cdot \mathtt{s}|\!)_\tau = \{\langle 0,1\rangle\}$ while $(\!|\overline{\mathtt{t} \cdot \overline{\mathtt{s}}}|\!)_\tau = \{\langle 0,0\rangle, \langle 0,1\rangle, \langle 1,1\rangle\}$. We have reached a contradiction, and conclude that $\overline{\mathtt{t}} \cdot \mathtt{s} \not\equiv \overline{\mathtt{t} \cdot \overline{\mathtt{s}}}$.

We can play the same game with guarded rational expressions. For instance, it is not too hard to show that an **if-then-else** construct admits a kind of right-distributivity: for all $e, f, g \in \mathbb{G}$ and $b \in \mathbb{B}$, it holds that

$$(\textbf{if } b \textbf{ then } e \textbf{ else } f) \cdot g \equiv \textbf{if } b \textbf{ then } e \cdot g \textbf{ else } f \cdot g$$

This may lead you to wonder: is there also a left-distributivity? More precisely, let's take $\mathtt{a}, \mathtt{b}, \mathtt{c} \in \Sigma$ and $\mathtt{t} \in T$. Can we show that the following holds:

$$\mathtt{a} \cdot (\textbf{if } \mathtt{t} \textbf{ then } \mathtt{b} \textbf{ else } \mathtt{c}) \equiv \textbf{if } \mathtt{t} \textbf{ then } \mathtt{a} \cdot \mathtt{b} \textbf{ else } \mathtt{a} \cdot \mathtt{c} \quad ?$$

Looking at this, you may already notice that something is intuitively wrong. On the left, the action a is executed before the test t, while on the right t is run first. So, from a programming perspective, it could well be the case that t succeeds before a, but fails afterwards. This is what we will base our countermodel on:

$$S = \{0,1,2\} \qquad \tau(\mathtt{t}) = \{0\} \qquad \sigma(\mathtt{a}) = \{\langle n, 0\rangle : n \in S\}$$

$$\sigma(\mathtt{b}) = \{\langle n, 1\rangle : n \in S\} \qquad\qquad \sigma(\mathtt{c}) = \{\langle n, 2\rangle : n \in S\}$$

Under this interpretation, the programs have different semantics; specifically:

$$\langle 1,1\rangle \in [\![\mathtt{a} \cdot (\textbf{if } \mathtt{t} \textbf{ then } \mathtt{b} \textbf{ else } \mathtt{c})]\!]_{\sigma,\tau} \qquad \langle 1,1\rangle \not\in [\![\textbf{if } \mathtt{t} \textbf{ then } \mathtt{a} \cdot \mathtt{b} \textbf{ else } \mathtt{a} \cdot \mathtt{c}]\!]_{\sigma,\tau}$$

Of course, it may happen that your equality is true in *some* models, but not in others. For instance, if we choose $S = T$, $\tau(\mathtt{t}) = \{\mathtt{t}\}$ and $\tau(\mathtt{s}) = \{\mathtt{s}\}$, then $(\!|\overline{\mathtt{t}} \cdot \mathtt{s}|\!)_\tau = (\!|\overline{\mathtt{t} \cdot \overline{\mathtt{s}}}|\!)_\tau$, despite $\overline{\mathtt{t}} \cdot \mathtt{s} \not\equiv \overline{\mathtt{t} \cdot \overline{\mathtt{s}}}$, as we have seen. Indeed, in the extreme, we can simply choose the null model, $S = \emptyset$, $\tau(-) = \emptyset$ and $\sigma(-) = \emptyset$, which satisfies *any* equivalence. It should therefore be clear that, if you want to invalidate an equation using a relational model, you should put some intuition and thought in your choices, as we have done above.

3 A language model

As we've seen, refutation using a relational model is possible, but requires some creativity, especially in the choice of your semantic domain. Maybe you're not feeling creative, or maybe you would rather automate the search for a countermodel. In that case, it's useful to have a different kind of model, one where you do not have to make as many choices, instead simply having to compute.

That's where the *language model* comes in. Intuitively, the language model of rational expressions can be seen as listing all possible ways in which the primitive actions in the expression can be executed. Let's develop some theory.

We write Σ^* for the set of *words* over Σ, i.e., the set of finite sequences $\mathtt{a}_0\mathtt{a}_1\cdots\mathtt{a}_{n-1}$ where $\mathtt{a}_i \in \Sigma$. We write ϵ for the empty word. When $w, x \in \Sigma^*$, we write wx for their *concatenation*, i.e., the letters of w followed by the letters from x. A set of words (over Σ) is called a *language* (over Σ). Concatenation can be lifted: when L and K are languages, we write $L \cdot K$ for the language $\{wx : w \in L, x \in K\}$, and L^* for the *Kleene closure*: $\{w_0w_1\cdots w_{n-1} : w_i \in L\}$.

We now have everything we need to give the language model of rational expressions. Specifically, we define $[\![-]\!]_{\mathbb{E}} : \mathbb{E} \to 2^{\Sigma^*}$ inductively, as follows:

$$[\![0]\!]_{\mathbb{E}} = \emptyset \qquad [\![1]\!]_{\mathbb{E}} = \{\epsilon\} \qquad [\![\mathtt{a}]\!]_{\mathbb{E}} = \{\mathtt{a}\}$$

$$[\![e+f]\!]_{\mathbb{E}} = [\![e]\!]_{\mathbb{E}} \cup [\![f]\!]_{\mathbb{E}} \qquad [\![e\cdot f]\!]_{\mathbb{E}} = [\![e]\!]_{\mathbb{E}} \cdot [\![f]\!]_{\mathbb{E}} \qquad [\![e^*]\!]_{\mathbb{E}} = [\![e]\!]_{\mathbb{E}}^*$$

Every word in $[\![e]\!]_{\mathbb{E}}$ represents a sequence of actions that could be executed by e: for instance, $[\![1]\!]_{\mathbb{E}}$ contains only the empty word, because the program 1 does not execute any actions. Similarly, $[\![e \cdot f]\!]_{\mathbb{E}}$ first lists a sequence of actions from e, and then a sequence of actions from f, and likewise for the other operators.

As an example of the language semantics, consider the rational expression $\mathtt{a} \cdot (\mathtt{b} \cdot \mathtt{a})^*$. Its language semantics is the set of all words of the form $\mathtt{aba}\cdots\mathtt{ba}$. As you've seen before, this expression is provably equivalent to $(\mathtt{a} \cdot \mathtt{b})^* \cdot \mathtt{a}$, and indeed, the language semantics of this expression is the same. Just like the relational semantics that we saw before, this property holds in general:

Lemma 3.2 (Language soundness). *Let $e, f \in \mathbb{E}$. If $e \equiv f$, then $[\![e]\!]_{\mathbb{E}} = [\![f]\!]_{\mathbb{E}}$.*

Proof. As before, the proof proceeds by induction on the construction of $\equiv$. In the base, we verify each of the rules. For example, since $e \cdot 0 \equiv 0$, we should check that $[\![e \cdot 0]\!]_{\mathbb{E}} = [\![0]\!]_{\mathbb{E}}$. Luckily, this is the case: both languages are empty.

For the inductive step, we need to verify the congruence rules (these are easy), as well as the least fixpoint axiom $e + f \cdot g \leqq g \implies f^* \cdot e \leqq g$ and its symmetric cousin. If $e + f \cdot g \leqq g$, then $[\![e + f \cdot g]\!]_{\mathbb{E}} \subseteq [\![g]\!]_{\mathbb{E}}$ by induction. Now suppose that $w \in [\![f^* \cdot e]\!]_{\mathbb{E}}$. Then $w = w_0 \cdots w_{n-1}x$ where $w_0, \ldots, w_{n-1} \in [\![f]\!]_{\mathbb{E}}$ and $x \in [\![e]\!]_{\mathbb{E}}$. We can then show, by induction on n, that $w \in [\![g]\!]_{\mathbb{E}}$. The proof for the other least fixpoint axiom is similar. □

As hinted before, one of the advantages of the language model is that it saves you from having to choose an interpretation of the letters. For instance, if you want to show that $\mathtt{a} + \mathtt{b} \not\equiv \mathtt{a} \cdot \mathtt{b}$, it suffices to note that $[\![\mathtt{a} + \mathtt{b}]\!]_{\mathbb{E}} = \{\mathtt{a}, \mathtt{b}\}$ while $[\![\mathtt{a} \cdot \mathtt{b}]\!]_{\mathbb{E}} = \{\mathtt{ab}\}$, hence $[\![\mathtt{a}+\mathtt{b}]\!]_{\mathbb{E}} \neq [\![\mathtt{a} \cdot \mathtt{b}]\!]_{\mathbb{E}}$, and contrapositively apply Lemma 3.2.

4 A guarded language model

We can also create a similar model for guarded rational expressions. The idea here is to keep track of the "state" of the machine, in addition to the possible actions performed. Since our programming language can perform primitive tests from T, it stands to reason that the state of our machine is completely described by the set of primitive tests that succeed. Let's call such a set an *atom*. By extension, we can assign to each test b the set of atoms $[\![b]\!]_{\mathbb{B}} \subseteq 2^T$, as follows:

$$[\![0]\!]_{\mathbb{B}} = \emptyset \qquad [\![1]\!]_{\mathbb{B}} = 2^T \qquad [\![\mathtt{t}]\!]_{\mathbb{B}} = \{\alpha \subseteq T : \mathtt{t} \in \alpha\}$$

$$[\![b + c]\!]_{\mathbb{B}} = [\![b]\!]_{\mathbb{B}} \cup [\![c]\!]_{\mathbb{B}} \qquad [\![b \cdot c]\!]_{\mathbb{B}} = [\![b]\!]_{\mathbb{B}} \cap [\![c]\!]_{\mathbb{B}} \qquad [\![\bar{b}]\!]_{\mathbb{B}} = 2^T \setminus [\![b]\!]_{\mathbb{B}}$$

This is a sound semantics of Boolean expressions:

Lemma 3.3 (Boolean soundness)**.** *Let $b, c \in \mathbb{B}$. If $b \equiv_{\mathbb{B}} c$, then $[\![b]\!]_{\mathbb{B}} = [\![c]\!]_{\mathbb{B}}$.*

Proof sketch. As usual, we proceed by induction on the construction of $\equiv_{\mathbb{B}}$. In the base, we check the rules, such as $b + (c \cdot d) \equiv (b + c) \cdot (b + d)$. This means that we should check that $[\![b]\!]_{\mathbb{B}} \cup ([\![c]\!]_{\mathbb{B}} \cap [\![d]\!]_{\mathbb{B}}) = ([\![b]\!]_{\mathbb{B}} \cup [\![c]\!]_{\mathbb{B}}) \cap ([\![b]\!]_{\mathbb{B}} \cup [\![d]\!]_{\mathbb{B}})$, which can be verified using a straightforward argument.

The inductive step is even simpler than before, because we have no rules that involve an implication; all we need to check are the rules for congruence. For instance, if $b_0 \cdot c_0 \equiv b_1 \cdot c_1$ because $b_0 \equiv_{\mathbb{B}} b_1$ and $c_0 \equiv_{\mathbb{B}} c_1$, then by induction we know that $[\![b_0]\!]_{\mathbb{B}} = [\![b_1]\!]_{\mathbb{B}}$ and $[\![c_0]\!]_{\mathbb{B}} = [\![c_1]\!]_{\mathbb{B}}$, and hence we derive

$$[\![b_0 \cdot c_0]\!]_{\mathbb{B}} = [\![b_0]\!]_{\mathbb{B}} \cap [\![c_0]\!]_{\mathbb{B}} = [\![b_1]\!]_{\mathbb{B}} \cap [\![c_1]\!]_{\mathbb{B}} = [\![b_1 \cdot c_1]\!]_{\mathbb{B}}$$ □

It turns out that, as a matter of fact, the converse to the above also holds: if $[\![b]\!]_{\mathbb{B}} = [\![c]\!]_{\mathbb{B}}$, then $b \equiv_{\mathbb{B}} c$. The proof is a bit more involved, but certainly doable. We will discuss an analogous property for the semantics $[\![-]\!]_{\mathbb{E}}$ next week.

For now, we return to our more immediate objective of creating a language model for guarded rational expressions. Remember that the idea was to keep track of the actions performed by the machine, as well as the states that the machine was in. More precisely, our semantics be in terms of *guarded languages*, which are sets of words of the form $\alpha_0 \mathtt{a}_0 \alpha_1 \cdots \alpha_{n-1} \mathtt{a}_{n-1} \alpha_n$, where $\alpha_0, \dots, \alpha_n \in 2^T$, and $\mathtt{a}_0, \dots, \mathtt{a}_{n-1} \in \Sigma$ — in other words, subsets of the language $2^T \cdot (\Sigma \cdot 2^T)^*$.

The intuition is that such a *guarded word* in the semantics of $e \in \mathbb{G}$ records a possible way of executing e: α_0 is the initial state of the machine, $\mathtt{a}_0$ is the first action performed, α_1 is the state after that action, and so on until the final state α_n. In particular, note that an atom α is also a guarded word (of just one letter); such a word represents the execution of a program that starts in a state described by α, but does not perform any action, thus ending in the same state.

We can build a semantics in terms of guarded languages, as follows:

$$[\![b]\!]_{\mathbb{G}} = [\![b]\!]_{\mathbb{B}} \qquad [\![\mathtt{a}]\!]_{\mathbb{G}} = \{\alpha \mathtt{a} \beta : \alpha, \beta \in 2^T\} \qquad [\![e + f]\!]_{\mathbb{G}} = [\![e]\!]_{\mathbb{G}} \cup [\![f]\!]_{\mathbb{G}}$$

$$[\![e \cdot f]\!]_{\mathbb{G}} = \{w \alpha x : w\alpha \in [\![e]\!]_{\mathbb{G}}, \alpha x \in [\![f]\!]_{\mathbb{G}}\} \qquad [\![e^*]\!]_{\mathbb{G}} = \bigcup_{n \in \mathbb{N}} [\![\underbrace{e \cdots e}_{n \text{ times}}]\!]_{\mathbb{G}}$$

As before, we obtain a soundness result for this model as well:

Lemma 3.4 (Soundness for guarded languages)**.** *If $e \equiv_{\mathbb{G}} f$, then $[\![e]\!]_{\mathbb{G}} = [\![f]\!]_{\mathbb{G}}$.*

Proof sketch. The proof follows the same familiar pattern: we proceed by induction on $\equiv_{\mathbb{G}}$. In the base, we need to verify rules such as $e \cdot (f \cdot g) \equiv (e \cdot f) \cdot g$. Semantically, this means checking the (cumbersome but doable) equality

$$\begin{aligned}&\{y\beta z : y\beta \in [\![e]\!]_{\mathbb{G}}, \beta z \in \{w\alpha x : w\alpha \in [\![f]\!]_{\mathbb{G}}, \alpha x \in [\![g]\!]_{\mathbb{G}}\}\} \\ &\quad = \{y\beta z : y\beta \in \{w\alpha x : w\alpha \in [\![e]\!]_{\mathbb{G}}, \alpha x \in [\![f]\!]_{\mathbb{G}}\}, \beta z \in [\![g]\!]_{\mathbb{G}}\}\end{aligned}$$

The inductive step, where we verify the fixpoint rules as well as the congruence rules, can be completed using the same techniques we saw before. □

To drive home the point that we can use guarded languages in refutation, consider the two programs we compared earlier on: $\mathtt{a} \cdot (\textbf{if } \mathtt{t} \textbf{ then } \mathtt{b} \textbf{ else } \mathtt{c})$ and $\textbf{if } \mathtt{t} \textbf{ then } \mathtt{a} \cdot \mathtt{b} \textbf{ else } \mathtt{a} \cdot \mathtt{c}$. Computing their language semantics yields:

$$
\begin{aligned}
[\![\mathtt{a} \cdot (\textbf{if } \mathtt{t} \textbf{ then } \mathtt{b} \textbf{ else } \mathtt{c})]\!]_{\mathbb{G}} = \{&\emptyset\mathtt{a}\emptyset\mathtt{c}\emptyset, \emptyset\mathtt{a}\emptyset\mathtt{c}\{\mathtt{t}\}, \\
&\emptyset\mathtt{a}\{\mathtt{t}\}\mathtt{b}\emptyset, \emptyset\mathtt{a}\{\mathtt{t}\}\mathtt{b}\{\mathtt{t}\}, \\
&\{\mathtt{t}\}\mathtt{a}\emptyset\mathtt{c}\emptyset, \{\mathtt{t}\}\mathtt{a}\emptyset\mathtt{c}\{\mathtt{t}\}, \\
&\{\mathtt{t}\}\mathtt{a}\{\mathtt{t}\}\mathtt{b}\emptyset\}, \{\mathtt{t}\}\mathtt{a}\{\mathtt{t}\}\mathtt{b}\{\mathtt{t}\}\} \\
[\![\textbf{if } \mathtt{t} \textbf{ then } \mathtt{a} \cdot \mathtt{b} \textbf{ else } \mathtt{a} \cdot \mathtt{c}]\!]_{\mathbb{G}} = \{&\emptyset\mathtt{a}\emptyset\mathtt{c}\emptyset, \emptyset\mathtt{a}\emptyset\mathtt{c}\{\mathtt{t}\}, \\
&\emptyset\mathtt{a}\{\mathtt{t}\}\mathtt{c}\emptyset, \emptyset\mathtt{a}\{\mathtt{t}\}\mathtt{c}\{\mathtt{t}\}, \\
&\{\mathtt{t}\}\mathtt{a}\emptyset\mathtt{b}\emptyset, \{\mathtt{t}\}\mathtt{a}\emptyset\mathtt{b}\{\mathtt{t}\}, \\
&\{\mathtt{t}\}\mathtt{a}\{\mathtt{t}\}\mathtt{b}\emptyset, \{\mathtt{t}\}\mathtt{a}\{\mathtt{t}\}\mathtt{b}\{\mathtt{t}\}\}
\end{aligned}
$$

As you can tell, these sets are different; for instance, the first set contains $\emptyset\mathtt{a}\{\mathtt{t}\}\mathtt{b}\emptyset$, while the second does not. Even though the computation may be a bit more tedious, we did not have to come up with a suitable interpretation of the primitive tests and letters, while still reaching the same conclusion.

5 Equivalence of models

So far, we have seen that we can use both the relational and language interpretation to give a countermodel to a proclaimed equivalence. This raises the question: are these two methods equivalent? More precisely, can there be a relational countermodel of two terms that have the same language? Or, conversely, is it possible for two terms without a relational countermodel to have different languages? It turns out that the answer to both these questions is *no*. Let's start by answering the first question.

Theorem 3.5. *Let $e, f \in \mathbb{E}$. if $[\![e]\!]_{\mathbb{E}} = [\![f]\!]_{\mathbb{E}}$, then $[\![e]\!]_\sigma = [\![f]\!]_\sigma$ for all σ.*

Proof sketch. The idea is as follows: given an interpretation $\sigma : \Sigma \to 2^{S \times S}$, we create $\hat{\sigma} : 2^{\Sigma^*} \to 2^{S \times S}$, such that $[\![-]\!]_\sigma = \hat{\sigma}([\![-]\!]_{\mathbb{E}})$. The claim follows, since

$$[\![e]\!]_\sigma = \hat{\sigma}([\![e]\!]_{\mathbb{E}}) = \hat{\sigma}([\![f]\!]_{\mathbb{E}}) = [\![f]\!]_\sigma$$

As it turns out, the definition of $\hat{\sigma}$ is fairly simple:

$$\hat{\sigma}(L) = \{\sigma(\mathtt{a}_0) \circ \cdots \circ \sigma(\mathtt{a}_{n-1}) : \mathtt{a}_0 \cdots \mathtt{a}_{n-1} \in L\}$$

To verify the claimed equality, i.e., that $[\![g]\!]_\sigma = \hat{\sigma}([\![g]\!]_{\mathbb{E}})$ for all $g \in \mathbb{E}$, we can proceed by induction on g. In the base, where $g \in \{0, 1\} \cup \Sigma$, the claim is straightforward. For the inductive steps, it suffices to verify that $\hat{\sigma}$ is compatible with the operators of language composition, that is to say:

$$\hat{\sigma}(L \cup K) = \hat{\sigma}(L) \cup \hat{\sigma}(K) \qquad \hat{\sigma}(L \cdot K) = \hat{\sigma}(L) \circ \hat{\sigma}(K) \qquad \hat{\sigma}(L^*) = \hat{\sigma}(L)^*$$

These all turn out to hold; you will check one of them in the homework. □

The above tells us that *if some relational interpretation offers a counterexample to an equivalence, then so does the language model.* Conversely, if two terms have the same language, there is no hope of a relational countermodel.

Theorem 3.6. *Let $e, f \in \mathbb{E}$. If $[\![e]\!]_\sigma = [\![f]\!]_\sigma$ for all σ, then $[\![e]\!]_\mathbb{E} = [\![f]\!]_\mathbb{E}$.*

Proof. Consider the map $\sharp : 2^{\Sigma^*} \to 2^{\Sigma^* \times \Sigma^*}$, given by

$$'\sharp(L) = \{\langle w, wx\rangle : w \in \Sigma^*, x \in L\}$$

One can show that $\sharp$ is injective, i.e., that if $\sharp(L) = \sharp(K)$, then $L = K$. After all, if $w \in L$, then $\langle \epsilon, w\rangle \in \sharp(L) = \sharp(K)$, hence $w \in K$, and vice versa. By extension, this means that it suffices to show $\sharp([\![e]\!]_\mathbb{E}) = \sharp([\![f]\!]_\mathbb{E})$.

To this end, we choose the interpretation $\langle S, \sigma\rangle$, where

$$S = 2^{\Sigma^* \times \Sigma^*} \qquad \sigma(\mathtt{a}) = \sharp(\{\mathtt{a}\})$$

You will show (in the homework) that $[\![g]\!]_\sigma = \sharp([\![g]\!]_\mathbb{E})$ for all $g \in \mathbb{E}$, by induction on g. Essentially, this comes down to verifying the following equalities:

$$\sharp(L \cup K) = \sharp(L) \cup \sharp(K) \qquad \sharp(L \cdot K) = \sharp(L) \circ \sharp(K) \qquad \sharp(L^*) = \sharp(L)^*$$

By the premise, we know that for our choice of σ, we have $[\![e]\!]_\sigma = [\![f]\!]_\sigma$, and hence $\sharp([\![e]\!]_\mathbb{E}) = \sharp([\![f]\!]_\mathbb{E})$. By injectivity of $\sharp$, we conclude that $[\![e]\!]_\mathbb{E} = [\![f]\!]_\mathbb{E}$. □

From the above, we learn that *if the language model offers a counterexample to an equivalence, then so does some relational interpretation.* Conversely, if you have shown that there cannot be a relational countermodel to an equivalence, then you do not need to check the language model anymore.

The above theorems also hold when comparing guarded rational languages to the guarded relational interpretation; indeed, their proofs are almost the same. For the sake of brevity, we do not reproduce them here.

6 Looking ahead

So far, we have used inequivalence in some sound model as a *sufficient* condition to establish the absence of a proof. However, at this point, we have not shown *necessity*, i.e., that if $e \not\equiv f$, also $[\![e]\!]_\mathbb{E} \neq [\![f]\!]_\mathbb{E}$. This is not surprising, since the latter claim is equivalent to saying that if $[\![e]\!]_\mathbb{E} = [\![f]\!]_\mathbb{E}$, also $e \equiv f$ — in other words, *every semantic equivalence is provable.* It turns out that this is indeed true. The proof requires some machinery that we will develop next week.

Lastly, a word of warning: the equivalences that we considered so far are all strictly equational, i.e., of the form $e \equiv f$. More generally, one can consider *Horn equations*, of the form $e_0 \equiv f_0, \dots, e_{n-1} \equiv f_{n-1} \implies e \equiv f$. While some of these hold in the language model of rational expressions — in the sense that if $[\![e_0]\!]_\mathbb{E} = [\![f_0]\!]_\mathbb{E}, \dots, [\![e_{n-1}]\!]_\mathbb{E} = [\![f_{n-1}]\!]_\mathbb{E}$, then $[\![e]\!]_\mathbb{E} = [\![f]\!]_\mathbb{E}$ — this does not mean that they are also provable, even with the foreshadowed result. Indeed, there exist Horn equations that are true in the language model, but not in another (artificial, but sound) semantics of rational expressions.

7 Homework

1. Give a relational countermodel to prove that $(\mathtt{a} + \mathtt{b})^* \not\equiv \mathtt{a}^* \cdot (\mathtt{b} \cdot \mathtt{a})^*$ (note how the second term is slightly different from the equivalence that you have shown before, where the last $\mathtt{a}$ appeared below a star).

2. Use the language semantics to prove that $(\mathtt{a} + \mathtt{b})^* \not\equiv \mathtt{a}^* \cdot (\mathtt{b} \cdot \mathtt{a})^*$. Note that, since both languages are infinite, you will need to demonstrate (formally!) that some word occurs in one language, but not another.

   When you're done, think for a minute: which approach did you like best? No need to write down your thoughts.

3. Prove the three equalities claimed at the end of Theorem 3.5. Next, formally verify that, for all $g \in \mathbb{E}$, it holds that $[\![g]\!]_\sigma = \hat{\sigma}([\![g]\!]_\mathbb{E})$, by induction on g, and using those three equalities.

4. Prove the three equalities claimed at the end of Theorem 3.6. Next, formally verify that, for all $g \in \mathbb{E}$ and the choice of $\langle S, \sigma \rangle$ made there, it holds that $[\![g]\!]_\sigma = \sharp([\![g]\!]_\mathbb{E})$, by induction on g, and using those three equalities.

8 Bibliographical notes

The language model of Kleene algebra is as old as Kleene's original paper [Kle56]. In fact, the presentation in this course is a bit fictive — Kleene proposed studying the languages deriving from rational expressions; the connection to relational algebra was made later on. The guarded language model for guarded rational expressions is due to Kozen and Smith [KS96].

The correspondence between the language and relational model was first observed by Pratt [Pra80]. The analogous correspondence between guarded languages and relational models is also due to Kozen and Smith [KS96].

References

[Kle56] Stephen C. Kleene. Representation of events in nerve nets and finite automata. In Claude E. Shannon and John McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, 1956.

[KS96] Dexter Kozen and Frederick Smith. Kleene algebra with tests: Completeness and decidability. In *CSL*, pages 244–259, 1996. `doi:10.1007/3-540-63172-0_43`.

[Pra80] Vaughan R. Pratt. Dynamic algebras and the nature of induction. In *STOC*, pages 22–28, 1980. `doi:10.1145/800141.804649`.