

A Symbolic Decision Procedure for GKAT

Abstract

1 Symbolic Derivatives

Given a finite set K , a *symbolic Deterministic Kleene Coalgebra with Tests* (sDKCT) $\mathcal{S} \triangleq \langle S, \epsilon, \delta \rangle$ over a boolean algebra \mathcal{B} consists of a state set S consists of a accepting boolean ϵ and a transition function δ :

$$\epsilon : S \rightarrow \mathcal{B}, \quad \delta : S \rightarrow S \rightarrow K \rightarrow \mathcal{B},$$

where for all states $s \in S$, all the booleans are “disjoint”; namely the conjunction of any two expression from the set $\epsilon(s) + \{\delta(s, s', p) \mid s' \in S, p \in K\}$ are false. The ordering on \mathcal{S} is defined pointwise, namely

$$\langle \epsilon_1, \delta_1 \rangle \leq \langle \epsilon_2, \delta_2 \rangle \text{ when } \forall s, s' \in S, \forall p \in K, \epsilon_1(s) \leq \epsilon_2(s) \text{ and } \delta_1(s, s', p) \leq \delta_2(s, s', p),$$

where the ordering on the right hand side is defined in the boolean algebra.

Intuitively, the elements of boolean is treated as the “symbolic” transitions, which can be think of as a set of labels by the classical stone’s representation theorem. In particular, when the boolean algebra is the free boolean algebra \mathbf{BExp}_B generated by a finite set B , the labels can be considered as atoms, since $\mathbf{BExp}_B \cong 2^{\mathbf{At}_B}$. Then we can explain the intuition for the definition of DKCT

- $\epsilon(s)$ can be thought of as a set of labels that is accepted by the state s .
- $\delta(s, s', p)$ denotes all the labels that can transition from s to s' while executing p .

Then the labels that is not in either operations are implicitly rejecting.

Given a sDKCT $\mathcal{S} \triangleq \langle S, \epsilon, \delta \rangle$ and a state $s \in S$, we use the notation $s \Rightarrow b$ to denote that $\epsilon(s) \geq b$, and we use the notation $s \xrightarrow{b|p} s'$ to denote that $\delta(s, s', p) \geq b$, where the ordering is the typical ordering in boolean algebra.

The symbolic derivatives for GKAT forms a DKCT where the states are represented by GKAT expression $\mathbf{GKAT}_{K,B}$, and the boolean algebra is the free boolean algebra \mathbf{BExp}_B over the test alphabet B :

$$\epsilon : \mathbf{GKAT}_{K,B} \rightarrow \mathbf{BExp}_B; \quad \delta : \mathbf{GKAT}_{K,B} \rightarrow \mathbf{GKAT}_{K,B} \rightarrow K \rightarrow \mathbf{BExp}_B.$$

the accepting map ϵ and transition map δ can be defined by the smallest maps that satisfy the following rules:

$$\begin{array}{c} \frac{}{b \Rightarrow b} \quad \frac{}{p \xrightarrow{1|p} 1} \quad \frac{e \Rightarrow a}{e +_b f \Rightarrow ba} \quad \frac{f \Rightarrow a}{e +_b f \Rightarrow \bar{b}a} \quad \frac{e \xrightarrow{a|p} e'}{e +_b f \xrightarrow{b \wedge a|p} e'} \quad \frac{f \xrightarrow{a|p} f'}{e +_b f \xrightarrow{\bar{b} \wedge a|p} f'} \\[10pt] \frac{e \Rightarrow a \quad f \Rightarrow b}{e \cdot f \Rightarrow a \wedge b} \quad \frac{e \Rightarrow a \quad f \xrightarrow{b|p} f'}{e \cdot f \xrightarrow{a \wedge b|p} f'} \quad \frac{e \xrightarrow{b|p} e'}{e \cdot f \xrightarrow{b|p} e' \cdot f} \quad \frac{}{e^{(b)} \Rightarrow \bar{b}} \quad \frac{e \xrightarrow{a|p} e'}{e^{(b)} \xrightarrow{b \wedge a|p} e' \cdot e^{(b)}} \end{array}$$

2 Implementing derivatives

Since the rules listed above is monotonic, that is if we enlarge the boolean in the premise, we also enlarge the boolean in the result. So we can simply pick the largest boolean on the premise, which is either $\epsilon(e)$ or $\delta(e, e', p)$. Since the disjunction of two boolean $a \vee b$ is the smallest boolean that is greater than both a and b , thus we can implement ϵ and δ by iteration through all the rules, and take the disjunction of all the possible booleans.

$$\begin{aligned}\epsilon(b) &\triangleq b & \epsilon(e +_b f) &\triangleq (b \wedge \epsilon(e)) \vee (\bar{b} \wedge \epsilon(f)) \\ \epsilon(q) &\triangleq 0 & \epsilon(e \cdot f) &\triangleq \epsilon(e) \wedge \epsilon(f) \\ \epsilon(e^{(b)}) &\triangleq \bar{b}\end{aligned}$$

However, the derivative function

$$\delta : \text{GKAT}_{K,B} \rightarrow \text{GKAT}_{K,B} \rightarrow K \rightarrow \text{BExp}_B,$$

poses challenges in the implementation, since $\text{GKAT}_{K,B}$ is infinite, so it will be impractical to search through all the expressions. However, we can treat the function δ extensionally:

$$\text{GKAT}_{K,B} \rightarrow K \rightarrow \text{BExp}_B \subseteq 2^{\text{GKAT}_{K,B} \times K \times \text{BExp}_B},$$

and then because the boolean expression don't overlap for each input, hence the boolean expression are necessarily unequal; we can model the set by a partial map, thus all the δ can be represented as a function to the following partial maps.

$$\begin{aligned}\delta : \text{GKAT}_{K,B} &\rightarrow (\text{BExp}_B \rightharpoonup \text{GKAT}_{K,B} \times K) \\ \delta(b) &\triangleq \{\} \\ \delta(q) &\triangleq \{1 \mapsto (1, p)\} \\ \delta(e +_b f) &\triangleq \{b \wedge a \mapsto (e', p) \mid a \mapsto (e', p) \in \delta(e)\} \cup \{\bar{b} \wedge a \mapsto (f', p) \mid a \mapsto (f', p) \in \delta(f)\} \\ \delta(e \cdot f) &\triangleq \{b \mapsto (e' \cdot f, p) \mid b \mapsto (e, p) \in \delta(e)\} \cup \{\epsilon(e) \wedge b \mapsto (f', p) \mid b \mapsto (f', p) \in \delta(f)\} \\ \delta(e^{(b)}) &\triangleq \{b \wedge a \mapsto (e' \cdot e^{(b)}, p) \mid a \mapsto (e', p) \in \delta(e)\}.\end{aligned}$$

FIXME: we still need to think about what representation do we want to go with, with this representation, you can also have duplicated K. We need a good example here of why don't we go with map. One good argument is that this representation is complex.

3 Normalized Bisimulation

Bisimilarity implies language semantical equivalence

$$s \sim t \implies \llbracket s \rrbracket = \llbracket t \rrbracket$$

, however the converse doesn't hold unless we are working in a *normal* GKAT coalgebra. It is well known that two states in a normal GKAT coalgebra if and only if they have the same language semantics

$$s \sim t \iff \llbracket s \rrbracket = \llbracket t \rrbracket.$$

However, instead of normalizing automaton, we can in fact, check for normality on the fly, this is important to design a derivatives based decision algorithm.

We will use a notion of normalized bisimulation, where we will enlarge the bisimulation by dead states, since all the dead states are trace equivalent. Operationally, when we find a discrepancy during

the bisimulation algorithm we will check whether the states causing the discrepancy are dead. We can prove the correctness of this algorithm by proving that they are equivalent to normalize and then computing bisimulation. Let G be the functor defining GKAT coalgebra:

$$G \triangleq (2 + (-) \times K)^{\mathbf{At}}.$$

Given two DKCT \mathcal{S}, \mathcal{T} , and a relation on these two expression $\sim \subseteq S \times T$, we can define $\approx \subseteq (2 + S \times K) \times (2 + T \times K)$ as the smallest relation generated by the following rules: for all $p, q \in K$ and states $s' \in S, t' \in T$:

BISIM-0	BISIM-1	BISIM-TRANS	NORM-LEFT	NORM-RIGHT	NORM-BOTH
$\frac{}{0 \approx 0}$	$\frac{}{1 \approx 1}$	$\frac{s' \sim t'}{(s', p) \approx (t', p)}$	$\frac{s' \text{ is dead}}{(s', p) \approx 0}$	$\frac{t' \text{ is dead}}{0 \approx (t', p)}$	$\frac{s', t' \text{ are both dead}}{(s', p) \approx (t', q)}$

Intuitively, \approx enlarges the relation \sim for *results* of a transition. for example, it allows transition to dead states to equivalent to rejection, and so on. This extension enables “on-the-fly normalization”, where to determine whether $s \approx t$, we can simply first check for bisimulation using the bisimulation rules; and if all failed, we can check the normalization rules.

We call the bisimulation algorithm with on-the fly normalization “*normalized bisimulation*”. This concept can be summed up nicely using diagrams. For a relation \sim , then there is a straight forward embedding $(2 + \sim \times K) \hookrightarrow \approx$: This embedding generates all the elements that is formed by the bisimulation rules of \approx , also note that \sim is not restricted to live states, but dead states as well; this means that if two dead states follows the bisimulation rules, then we don’t need to fall back to the normalization rules. Since detecting whether a state is dead requires iterating through all its predecessors, we can use this property to forgo dead state detection until all the bisimulation rules failed.

The aforementioned embedding $(2 + \sim \times K) \hookrightarrow \approx$ can be point-wise lifted to $G(\sim) \hookrightarrow (\mathbf{At} \rightarrow \approx)$, thus our normalized algorithm can be seen as finding a normalized bisimulation \sim that satisfy the following diagram:

$$\begin{array}{ccccc}
S & \xleftarrow{\pi_1} & \sim & \xrightarrow{\pi_2} & T \\
\delta_S \downarrow & & \exists! \downarrow & & \downarrow \delta_T \\
(2 + S \times K)^{\mathbf{At}} & \xleftarrow{\pi_1^{\mathbf{At}}} & (\sim)^{\mathbf{At}} & \xrightarrow{\pi_2^{\mathbf{At}}} & (2 + T \times K)^{\mathbf{At}}
\end{array}$$

Notice, unlike $G(\sim)$, the map $\sim \mapsto (\sim)^{\mathbf{At}}$ is not a functor, since it is only defined on relations.

Example 1. Let’s consider a example with a single primitive test $B \triangleq \{b\}$, thus atoms are defined as $\mathbf{At}_B = \{\{b\}, \emptyset\}$. We focus on two states in the bisimulation s and t . with their respective transition defined as follows

$$\begin{array}{ll}
\delta_S(s, \{b\}) \triangleq 0 & \delta_S(t, \{b\}) \triangleq (t', p) \\
\delta_S(s, \emptyset) \triangleq (s', p) & \delta_S(t, \emptyset) \triangleq (t', p)
\end{array}$$

It is quite clear s and t will not be bisimilar, however, there can be a normalized bisimulation relation between them, provided that $s' \sim t'$ and t' is dead:

$$\begin{array}{ccccc}
s & \xleftarrow{\pi_1} & (s, t) & \xrightarrow{\pi_2} & t \\
\delta_S \downarrow & & \exists! \downarrow & & \downarrow \delta_T \\
\{b\} \mapsto 0 & \xleftarrow{\pi_1^{\mathbf{At}}} & \{b\} \mapsto 0, (t', p) & \xrightarrow{\pi_2^{\mathbf{At}}} & \{b\} \mapsto (t', p) \\
\emptyset \mapsto (s', p) & \xleftarrow{\pi_1^{\mathbf{At}}} & \emptyset \mapsto (s', p), (t', p) & \xrightarrow{\pi_2^{\mathbf{At}}} & \emptyset \mapsto (t', p)
\end{array}$$

Notice that $0 \approx (t', p)$ because t' is dead, and $(s', p) \approx (t', p)$ because we assume that $s' \sim t'$. Furthermore, if both s' and t' are dead, we can establish $s' \approx t'$ using either the BISIM-TRANS rule or NORM-BOTH rule, and we are free to pick the more efficient check in our implementation.

Theorem 1 (Universality). *Given a normalized bisimulation relation $\sim \subseteq S \times T$*

$$\begin{array}{ccccc}
 S & \xleftarrow{\pi_1} & \sim & \xrightarrow{\pi_2} & T \\
 \delta_S \downarrow & & h \downarrow & & \downarrow \delta_T \\
 (2 + S \times K)^{\mathbf{At}} & \xleftarrow{\pi_1^{\mathbf{At}}} & (\sim)^{\mathbf{At}} & \xrightarrow{\pi_2^{\mathbf{At}}} & (2 + T \times K)^{\mathbf{At}}
 \end{array}$$

Then h is unique, and defined as follows:

$$\begin{aligned}
 h : \sim &\rightarrow \approx^{\mathbf{At}} \\
 h(s, t, \alpha) &\triangleq (\delta_S(s, \alpha), \delta_T(s, \alpha))
 \end{aligned}$$

Proof. The commutativity can be verified by computation. We only need to show uniqueness.

The above commutative diagram implies the commutativity of the following diagram:

$$\begin{array}{ccccc}
 & & \sim & & \\
 & \swarrow \delta_S \circ \pi_1 & \downarrow h & \searrow \delta_T \circ \pi_2 & \\
 & (2 + S \times K)^{\mathbf{At}} & (\sim)^{\mathbf{At}} & (2 + T \times K)^{\mathbf{At}} & \\
 & \nwarrow \pi_1^{\mathbf{At}} & \downarrow i & \nearrow \pi_2^{\mathbf{At}} & \\
 (2 + S \times K)^{\mathbf{At}} & \xleftarrow{\pi_1^{\mathbf{At}}} & (2 + S \times K)^{\mathbf{At}} \times (2 + T \times K)^{\mathbf{At}} & \xrightarrow{\pi_2^{\mathbf{At}}} & (2 + T \times K)^{\mathbf{At}}
 \end{array}$$

Since $(-)^{\mathbf{At}}$ is a right adjoint and right adjoints preserve limits, $(2 + S \times K)^{\mathbf{At}} \times (2 + T \times K)^{\mathbf{At}}$ is the product of $(2 + S \times K)^{\mathbf{At}}$ and $(2 + T \times K)^{\mathbf{At}}$. Hence, $h \circ i$ is the unique function to make the above diagram commute. Assume we can replace h with g while preserving the commutativity of the diagram:

$$\begin{aligned}
 i \circ h &= i \circ g & (2 + S \times K)^{\mathbf{At}} \times (2 + T \times K)^{\mathbf{At}} &\text{ is a product.} \\
 \Rightarrow h &= g & i &\text{ is an embedding, hence left cancelable.}
 \end{aligned}$$

□

Definition 1. *Given a DKCT $\mathcal{S} = \langle S, \delta \rangle$, a sub-DKCT $\mathcal{S}' = \langle S', \delta|_{S'} \rangle$ is a coagebra induced by $S' \subseteq S$, a sub-DKCT can be characterized as a pullback square:*

$$\begin{array}{ccc}
 S' & \xrightarrow{i} & S \\
 \downarrow \delta|_{S'} & & \downarrow \delta \\
 G(S') & \xrightarrow{i} & G(S)
 \end{array}$$

By the commutativity of this diagram i is also a DKCT homomorphism. A principle sub-DKCT of state $s \in S$, denoted $\langle s \rangle$ is the smallest sub-DKCT that contains s . We can show that the principle DKCT of s exactly characterize the DKCT induced by the reachable state of s .

Lemma 1. *A state s is live if and only if there exists a accepting state in $\langle s \rangle$; and a state s is dead if and only if there is no accepting state in $\langle s \rangle$.*

Corollary 1. *s is dead if and only if $\langle s \rangle$ is all dead.*

Proof. Take any state $s' \in \langle s \rangle$, we can construct the DKCT $\langle s' \rangle$. Since $\langle s' \rangle$ is the smallest sub-DKCT that contains s' , and $s' \in \langle s \rangle$, thus $\langle s' \rangle \subseteq \langle s \rangle$. Finally, because there is no accepting state in $\langle s \rangle$, there cannot be any accepting state in $\langle s' \rangle$ thus s' is also dead. \square

In fact the normalization operation commutes with principle sub-DKCT. This result is intuitive as taking all the reachable states and then normalize is the same as normalize and then take all the reachable state.

Lemma 2. *sub-DKCT preserves principle sub-DKCT, that is, given $\mathcal{S}' \sqsubseteq \mathcal{S}$ and $s \in \mathcal{S}'$, then*

$$\langle s \rangle_{\mathcal{S}'} = \langle s \rangle_{\mathcal{S}}.$$

Proof. First by definition, $s \in \langle s \rangle_{\mathcal{S}}$, then we show that $\langle s \rangle_{\mathcal{S}}$ is the smallest sub-DKCT of \mathcal{S}' , that contains s in \mathcal{S} :

Because $s \in \langle s \rangle_{\mathcal{S}'}$ and $\langle s \rangle_{\mathcal{S}'} \sqsubseteq \mathcal{S}' \sqsubseteq \mathcal{S}$, by minimality of $\langle s \rangle_{\mathcal{S}}$:

$$\langle s \rangle_{\mathcal{S}} \sqsubseteq \langle s \rangle_{\mathcal{S}'} \sqsubseteq \mathcal{S}',$$

that is, $\langle s \rangle_{\mathcal{S}}$ is a sub-DKCT of \mathcal{S}' .

Given any $\mathcal{T} \sqsubseteq \mathcal{S}'$ and $s \in \mathcal{T}$, we have $\mathcal{T} \sqsubseteq \mathcal{S}' \sqsubseteq \mathcal{S}$, thus by minimality of $\langle s \rangle_{\mathcal{S}}$: $\langle s \rangle_{\mathcal{S}} \sqsubseteq \mathcal{T}$. In other word, $\langle s \rangle_{\mathcal{S}}$ is the the minimal sub-algebra of \mathcal{S}' that contains s , reaching desired conclusion. \square

Lemma 3. *sub-DKCT preserves liveness, that is given $\mathcal{S}' \sqsubseteq \mathcal{S}$ and $s \in \mathcal{S}'$,*

$$s \text{ is live in } \mathcal{S}' \iff s \text{ is live in } \mathcal{S}$$

Proof. recall that sub-DKCT preserves principle sub-DKCT, namely $\langle s \rangle_{\mathcal{S}'} = \langle s \rangle_{\mathcal{S}}$, thus

$$\begin{aligned} s \text{ is live in } \mathcal{S}' &\iff \langle s \rangle_{\mathcal{S}'} \text{ contains an accepting state} \\ &\iff \langle s \rangle_{\mathcal{S}} \text{ contains an accepting state} \\ &\iff s \text{ is live in } \mathcal{S} \end{aligned}$$

\square

Lemma 4. *norm is a functor, and it preserves inclusion, that is*

$$\mathcal{S} \sqsubseteq \mathcal{T} \implies \text{norm}(\mathcal{S}) \sqsubseteq \text{norm}(\mathcal{T}).$$

Lemma 5. *Given a DKCT \mathcal{S} , and two sub-DKCT $\mathcal{S}_1 \sqsubseteq \mathcal{S}$ and $\mathcal{S}_2 \sqsubseteq \mathcal{S}$, $\mathcal{S}_1 = \mathcal{S}_2$ if and only if their carrier set is equal.*

Lemma 6. *Given a DKCT \mathcal{S} , its dead states induces a sub-DKCT.*

Theorem 2. *Given a DKCT \mathcal{S} and a live state $s \in \mathcal{S}$:*

$$\langle s \rangle_{\text{norm}(\mathcal{S})} = \text{norm}(\langle s \rangle_{\mathcal{S}}).$$

Proof. We need to show $\text{norm}(\langle s \rangle_{\mathcal{S}})$ is the smallest sub-DKCT in $\text{norm}(\mathcal{S})$ that contains s .

First, $\text{norm}(\langle s \rangle_{\mathcal{S}})$ contains s because s is live. Second, take any $\mathcal{T} \sqsubseteq \text{norm}(\mathcal{S})$, we will use $\overline{\mathcal{T}}$ to denote the smallest sub-DKCT of \mathcal{S} that contains all the states in \mathcal{T} . Since \mathcal{T} is a sub-DKCT of $\text{norm}(\mathcal{S})$, $\overline{\mathcal{T}} \setminus \mathcal{T}$ can only contain dead states, thus $\text{norm}(\overline{\mathcal{T}}) = \mathcal{T}$.

$$\begin{aligned} \langle s \rangle_{\mathcal{S}} &\sqsubseteq \overline{\mathcal{T}} & \overline{\mathcal{T}} &\text{ contains } s \\ \implies \text{norm}(\langle s \rangle_{\mathcal{S}}) &\sqsubseteq \text{norm}(\overline{\mathcal{T}}) & &\text{ monotonicity of norm} \\ \implies \text{norm}(\langle s \rangle_{\mathcal{S}}) &\sqsubseteq \mathcal{T} & &\text{ norm}(\overline{\mathcal{T}}) = \mathcal{T} \end{aligned}$$

\square

We will show this by establishing the equivalence between “normalized bisimulation” and “bisimulation in normalized coalgebra”. Given a GKAT coalgebra \mathcal{S} , we can normalize it via removing the dead states and reroute all the transitions to dead state as direct rejections. Similarly, any relation $\sim \subseteq \mathcal{S} \times \mathcal{T}$ can be normalized by filtering out all the states that are live:

$$\text{norm}(\sim) \triangleq \{(s, t) \mid \text{both } s \text{ and } t \text{ are live states}\}.$$

Thus there is a natural embedding $\text{norm}(\sim) \hookrightarrow \sim$ by inclusion. Similarly, there is a embedding $G(\text{norm}(\sim)) \rightarrow (\sim)^{\mathbf{At}}$ by lifting the following embedding point-wise via $(-)^{\mathbf{At}}$:

$$\begin{aligned} 2 + \text{norm}(\sim) \times K &\hookrightarrow \sim \\ 0 &\mapsto (0, 0) \\ 1 &\mapsto (1, 1) \\ (s, t), p &\mapsto (s, p), (t, p) \end{aligned}$$

These two embeddings allows us to construct the following pullback square:

$$\begin{array}{ccc} \text{norm}(\sim) & \xhookrightarrow{i} & \sim \\ \text{norm}(h) \downarrow & & \downarrow h \\ G(\text{norm}(\sim)) & \xhookrightarrow{i} & (\sim)^{\mathbf{At}} \end{array}$$

Theorem 3. *A normalized bisimulation induces a bisimulation in the normal coalgebras. That is*

$$\begin{array}{ccccc} S & \xleftarrow{\pi_1} & \sim & \xrightarrow{\pi_2} & T \\ \delta_S \downarrow & & (\delta_S, \delta_T) \downarrow & & \downarrow \delta_T \\ G(S) & \xleftarrow{\pi_1^{\mathbf{At}}} & \sim^{\mathbf{At}} & \xrightarrow{\pi_2^{\mathbf{At}}} & G(T) \end{array} \Rightarrow \begin{array}{ccccc} \text{norm}(S) & \xleftarrow{\pi_1} & \text{norm}(\sim) & \xrightarrow{\pi_2} & \text{norm}(T) \\ \delta_{\text{norm}(S)} \downarrow & & \text{norm}(h) \downarrow & & \downarrow \delta_{\text{norm}(T)} \\ G(\text{norm}(S)) & \xleftarrow{G(\pi_1)} & G(\text{norm}(\sim)) & \xrightarrow{G(\pi_2)} & G(\text{norm}(T)) \end{array}$$

Proof. We can □

Lemma 7. *Given a normalized bisimulation $\sim \subseteq \mathcal{S} \times \mathcal{T}$, and a pair of state $s \sim t$, then the range of $\pi_1 : \sim \rightarrow S$, will cover all the live states in $\langle s \rangle$; similarly for the state $t \in \mathcal{T}$.*

Proof. let $\text{norm}(\sim)|_{\langle s \rangle}$ be all the live elements of \sim restricted to $\langle s \rangle$:

$$\text{norm}(\sim)|_{\langle s \rangle} \triangleq \{(s', t') \in \sim \mid s' \in \langle s \rangle, \text{both } s' \text{ and } t' \text{ are live}\}.$$

since all the live states in $\langle s \rangle$ is the carrier of $\text{norm}(\langle s \rangle)$, we only need to show $\pi_1 : \text{norm}(\sim)|_{\langle s \rangle} \rightarrow \langle s \rangle$ is surjective.

Since \sim is a normalize bi-simulation, then $\text{norm}(\sim)$ is a bisimulation, and the following diagram commute by computation:

$$\begin{array}{ccc} \text{norm}(\sim)|_{\langle s \rangle} & \xrightarrow{\pi_1} & \text{norm}(\langle s \rangle) \\ h \downarrow & & \downarrow \delta_{\text{norm}(S)} \\ G(\text{norm}(\sim)|_{\langle s \rangle}) & \xrightarrow{G(\pi_1)} & G(\text{norm}(\langle s \rangle)) \end{array}$$

Because of above diagram, $\mathbf{Img}(\pi_1) \subseteq \text{norm}(\langle s \rangle)$, the range of π_1 , also forms a sub-DKCT of $\text{norm}(S)$ with the transition: $\delta_{\text{norm}(S)} : \mathbf{Img}(\pi_1) \rightarrow G(\mathbf{Img}(\pi_1))$ then there are two cases:

- if s is dead, then $\text{norm}(\langle s \rangle)$ is empty, thus π_1 is surjective,

- if s is live, then $\text{norm}(\langle s \rangle) = \langle s \rangle_{\text{norm}(\mathcal{S})}$, because $\mathbf{Img}(\pi_1)$ is a sub-DKCT and contains s , thus

$$\text{norm}(\langle s \rangle) = \langle s \rangle_{\text{norm}(\mathcal{S})} \subseteq \mathbf{Img}(\pi_1).$$

Hence $\mathbf{Img}(\pi_1)$ covers $\text{norm}(\langle s \rangle)$ and π_1 is surjective.

□

Theorem 4. *Given a dead state s and a live state t , there doesn't exist a normalized bisimulation \sim s.t. $s \sim t$ or $t \sim s$.*

Proof. WLOG we will only prove that $s \sim t$ will lead to a contradiction. We can restrict the simulation diagram of \sim to $\langle s \rangle$ and $\text{norm}(\langle t \rangle)$, let

$$\sim_{s,t} = \{s', t' \mid s' \sim t', s' \in \langle s \rangle, t' \in \text{norm}(\langle t \rangle)\}.$$

Note that only \mathcal{T} is restricted to $\text{norm}(\langle t \rangle)$; while \mathcal{S} is restricted to $\langle s \rangle$, without norm. This discrepancy is because s is dead, thus $\text{norm}(\langle s \rangle)$ is empty. If we restrict \sim to $\text{norm}(\langle t \rangle) \times \text{norm}(\langle s \rangle)$, then there will be no element in the restriction, since $\text{norm}(\langle t \rangle) \times \text{norm}(\langle s \rangle)$ is empty.

We can show that $\sim_{s,t}$ form a normalized bisimulation between $\langle s \rangle$ and $\text{norm}(\langle t \rangle)$.

$$\begin{array}{ccccc} \langle s \rangle & \xleftarrow{\pi_1} & \sim_{s,t} & \xrightarrow{\pi_2} & \text{norm}(\langle t \rangle) \\ \delta_s \downarrow & & \downarrow (\delta_s, \text{norm}(\delta_{\mathcal{T}})) & & \downarrow \text{norm}(\delta_{\mathcal{T}}) \\ G(\langle s \rangle) & \xleftarrow{\pi_1 \mathbf{A}t} & \sim_{s,t} \mathbf{A}t & \xrightarrow{\pi_2 \mathbf{A}t} & G(\text{norm}(\langle t \rangle)) \end{array}$$

Then we can perform epi-mono factorization on the right square to obtain the range of π_2 :

$$\begin{array}{ccccc} \sim_{s,t} & \xrightarrow{\pi_2} & \mathbf{Img}(\pi_2) & \xhookrightarrow{i} & \text{norm}(\langle t \rangle) \\ (\delta_s, \text{norm}(\delta_{\mathcal{T}})) \downarrow & & \downarrow \text{norm}(\delta_{\mathcal{T}}) & & \downarrow \text{norm}(\delta_{\mathcal{T}}) \\ \sim_{s,t} \mathbf{A}t & \xrightarrow{\pi_2 \mathbf{A}t} & G(\mathbf{Img}(\pi_2)) & \xhookrightarrow{G(i)} & G(\text{norm}(\langle t \rangle)) \end{array}$$

Thus,

$$\mathbf{Img}(\pi_2) \subseteq \text{norm}(\langle t \rangle) \subseteq \text{norm}(\mathcal{T}).$$

Recall that $\text{norm}(\langle t \rangle_{\mathcal{T}}) = \langle t \rangle_{\text{norm}(\mathcal{T})}$, hence $\text{norm}(\langle t \rangle)$ is the smallest sub-DKCT of $\text{norm}(\mathcal{T})$, since $s \sim t$, thus $t \in \mathbf{Img}(\pi_2)$. Hence $\mathbf{Img}(\pi_2) \supseteq \text{norm}(\langle t \rangle)$, thus π_2 is surjective.

Since t is a live state, hence there exists an accepting state t' in $\text{norm}(\langle t \rangle)$; because s is a dead state, thus all the states in $\langle s \rangle$ is dead. Because $\pi_2 : \sim_{s,t} \rightarrow T$ is surjective, there exists a dead state $s' \in \langle s \rangle$ s.t. $s' \sim t'$. However, a dead state cannot be normally bisimilar to a accepting state, since $((s', p), 1) \notin \overline{\sim_{s,t}}$ for all dead state s' and action p . □

Theorem 5 (soundness and completeness). *Given a normalized bisimulation $\sim \subseteq \mathcal{S} \times \mathcal{T}$, then for all state $s \in \mathcal{S}$ and $t \in \mathcal{T}$, \sim is sound and complete with respect to trace equivalence*

$$s \sim t \text{ or both } s \text{ and } t \text{ are dead} \iff \llbracket s \rrbracket = \llbracket t \rrbracket.$$

Proof. The \implies direction can be shown by case analysis on the liveness of s and t :

- If both s and t are live,

$$s \sim t \implies (s, t) \in \text{norm}(\sim) \implies \llbracket s \rrbracket = \llbracket t \rrbracket.$$

- If one of s and t is live, and the other is dead, then $s \sim t$ cannot hold.
- If both s and t are dead, since dead states don't have any trace:

$$\llbracket s \rrbracket = \llbracket t \rrbracket = \emptyset.$$

The \Leftarrow direction can also be shown using similar case analysis:

- If both s and t are live, then by definition of $\text{norm}(\sim)$

$$\llbracket s \rrbracket = \llbracket t \rrbracket \implies (s, t) \in \text{norm}(\sim) \implies s \sim t.$$

- If one of s and t is live, and the other is dead, then one would contain traces, yet the other don't. Hence their trace semantics cannot equal.
- If both s and t are dead, we have reached our conclusion.

□

Corollary 2. *The trace semantics of a state is preserved in sub-DKCTs, that is given $\mathcal{S}' \sqsubseteq \mathcal{S}$ and a state $s \in \mathcal{S}'$, $\llbracket s \rrbracket_{\mathcal{S}} = \llbracket s \rrbracket_{\mathcal{S}'}$*

Proof. We only need to exhibit a normalized bisimulation between \mathcal{S}' and \mathcal{S} :

$$s \sim t \text{ when } s = t \text{ or } s \text{ and } t \text{ are both dead.}$$

To show that the relation \sim is a normalized bisimulation, we only need to check the commutativity of the following diagram:

$$\begin{array}{ccccc} \mathcal{S}' & \xleftarrow{\pi_1} & \sim & \xrightarrow{\pi_2} & \mathcal{S} \\ \delta_s \downarrow & & \downarrow (\delta_s, \delta_s) & & \downarrow \delta_s \\ G(\mathcal{S}') & \xleftarrow{\pi_1^{\mathbf{At}}} & \sim^{\mathbf{At}} & \xrightarrow{\pi_2^{\mathbf{At}}} & G(\mathcal{S}) \end{array}$$

The commutativity is apparent by computation. The function $(\delta_s, \delta_s) : \sim \rightarrow \sim^{\mathbf{At}}$ is well-defined, by case analysis on the input:

- if $s = t$, then $\delta_s(s, \alpha) = \delta_s(t, \alpha)$, hence will be in \sim for all α .
- if both s and t are dead, then for all $\alpha \in \mathbf{At}$ both $\delta_s(s, \alpha)$ and $\delta_s(t, \alpha)$ are will either transition to a dead state or reject, thus the results will also be in \sim .

□

Finally, since sub-DKCT preserves semantics, we can restrict \sim to $\langle s \rangle$ and $\langle t \rangle$ to minimize the search space. We denote the restricted normalized bisimulation as $\sim_{s,t}$:

$$\begin{array}{ccccc} \langle s \rangle & \xleftarrow{\pi_1} & \sim_{s,t} & \xrightarrow{\pi_2} & \langle t \rangle \\ \delta_s \downarrow & & \downarrow (\delta_s, \delta_{\mathcal{T}}) & & \downarrow \delta_{\mathcal{T}} \\ G(\langle s \rangle) & \xleftarrow{\pi_1^{\mathbf{At}}} & \sim_{s,t}^{\mathbf{At}} & \xrightarrow{\pi_2^{\mathbf{At}}} & G(\langle t \rangle) \end{array}$$

The because sub-DKCT preserves trace semantics, thus the completeness result still holds:

$$s \sim_{s,t} t \text{ or both } s \text{ and } t \text{ are dead} \iff \llbracket s \rrbracket = \llbracket t \rrbracket,$$

Our algorithm will focus on finding \sim in a single DKCT with transition function δ . This approach is general enough, when given \mathcal{S} and \mathcal{T} , we can merge them via coproduct $\mathcal{S} + \mathcal{T}$, where both \mathcal{S} and \mathcal{T} are sub-DKCT of $\mathcal{S} + \mathcal{T}$, thus preserving the semantics of all the states.

Given a state $s \in \mathcal{S}$ we will run a simple depth-first search in $\langle s \rangle$ to determine whether s is dead. Specifically recall that s is live if and only if there exists an accepting state in s , and if s is dead, then all of $\langle s \rangle$ is dead

```
(** return whether the state is accepting*)
let is_accept (s: state): bool =
   $\exists \alpha \in \mathbf{At}, \delta(s, \alpha) = \text{Accept}$ 

(** a mutable set to keep track of states
that has been explored by check_dead function*)
let explored =  $\emptyset$ 

(** return a all the states in  $\langle s \rangle$  if  $s$  is dead,
otherwise return None*)
let check_dead (s: state): state set option =
  (*  $\langle s \rangle$  has all be explored *)
  if  $s \in \text{explored}$  then  $\emptyset$ 

  (* exploring  $\langle s \rangle$  *)
  else for  $\alpha \in \mathbf{At}$ :
    if  $\delta(s, \alpha) = \text{Accept}$  then None
    else if  $\delta(s, \alpha) = (s', p)$  then
      if check_dead s' = None then None
      else explored := explored  $\cup$  check_dead s'
    (*  $\delta(s, \alpha) = \text{Reject case, skip}$  *)

  (* finished exploring  $\langle s \rangle$  *)
  explored  $\cup$  {s}
```

To achieve better efficiency, we can cache all the dead state we found:

```
(** a mutable set to keep track of all the found dead states *)
let dead_states =  $\emptyset$ 

(** a function to check whether a state is dead *)
let is_dead (s: state): bool =
  if  $s \in \text{dead\_states}$  then true
  else if check_dead s = None then false
  else
    dead_states := dead_states  $\cup$  check_dead s
    return true
```

Then we can design our algorithm to construct \sim from two input... TODO: think a bit about the best way to check for both dead.