Transition System

Definition

Transition System

TS 是一种 reactive system

> It is used to describe the potential behavior of discrete systems. It consists of states and transitions between states, which may be labeled with labels chosen from a set; the same label may appear on more than one transition.

如果用图像来表示，TS 是一个二元图，node 代表 state，edge 代表 transition

更为正式的定义为

一个 TS 是一个 5-tuple $A = (S, S_0, T, \alpha, \beta)$，其中

- S 是 state set（可以是无限甚至不可数的）
- S_0 是初始的状态
- T 是 transition set（可以是无限甚至不可数的）
- α, β 是两个从 T 到 S 的映射，对于 $t \in T$，$\alpha(t), \beta(t)$ 代表其 source 和 target

如果 S, T 是 finite，则称这个 TS 是 finite

Reachable and Terminal

如果一个 transition t 的 source，target 分别是 s, s'，则可以记为 $t : s \to s'$

根据此给出更为 general 的 transition $\twoheadrightarrow$

Basis. 如果 $s \to s'$，则有 $s \twoheadrightarrow s'$

Induction. 如果 $s \twoheadrightarrow s', s' \twoheadrightarrow s''$，则有 $s \twoheadrightarrow s''$

根据此即可定义一个状态的**可达**（reachable）

对于 TS $A = (S, S_0, T, \alpha, \beta)$，如果对于 $s \in S, s_0 \in S_0$ 有 $s_0 \twoheadrightarrow s$，则称 s reachable

而如果对于 $s \in S$，不存在 s' 满足 $s \to s'$，则称 s 为 terminal

若 s 为 terminal 且 s reachable，则称 s 为 **deadlock state**

Path

一条长度为 $n(n>0)$ 的 path 是一个 transition 的序列 $t_1, t_2 \ldots, t_n$ 满足

$$\forall 1 \leqslant i < n, \beta(t_i) = \alpha(t_{i+1})$$
$$\alpha(t_1) = S_0$$

上述定义是 finite path，而一个 infinite path 是一个 infinite 的 transition 序列 $t_1, t_2 \ldots, t_n, \ldots$ 满足

$$\forall i \geqslant 1, \beta(t_i) = \alpha(t_{i+1})$$
$$\alpha(t_1) = S_0$$

若将所有 finite path 的集合记为 T^+，将所有 infinite path 的集合记为 T^ω，则 α, β 可以扩展到 T^+

$$\alpha(t_1 \ldots t_n) = \alpha(t_1)$$
$$\beta(t_1 \ldots t_n) = \beta(t_n)$$

同样的，α 可以扩展到 T^ω

$$\alpha(t_1 \ldots) = \alpha(t_1)$$

由此可以定义 T^+ 上的 product，如果 $c = t_1 \ldots t_n$ 是长度为 n 的 path，而 $c' = t'_1 \ldots t'_m$ 是长度为 m 的 path，且满足 $\beta(c) = \alpha(c')$，则可定义

$$c \cdot c' = t_1 \ldots t_n t'_1 \ldots t'_m$$

为长度为 $n+m$ 的 path，且 $\alpha(c \cdot c') = \alpha(c), \beta(c \cdot c') = \beta(c')$

product 也可推广到 $T^+ \times T^\omega$，定义与上述类似

Empty path：对于每个状态 s，都可以定义一条 empty path ϵ_s，满足

$$\alpha(\epsilon_s) = \beta(\epsilon_s) = s$$

则如果对于 path c 有 $\alpha(c) = s, \beta(c) = s'$，则 $\epsilon_s \cdot c = c = c \cdot \epsilon_{s'}$

Labeled transition system

A transition system labeled by an alphabet Σ is a 6-tuple
$A = (S, S_0, T, \alpha, \beta, \lambda)$

其中 $\lambda : T \to \Sigma$

label $\lambda(t)$ 代表触发 t 的动作

一般来说不会有两个 transition 有相同的 source，target 以及 label，即 TS 中的 transition 都是可区分的（$T \to S \times \Sigma \times S$ 是单射），但是两个 transition 有相同的 source 和 target 是合理的

在 labeled TS 中，有 path $c = t_1 t_2 \dots$，则称 $trace(c) = \lambda(t_1)\lambda(t_2)\dots$ 为 c 的**迹**（trace）

Equivalency for TS

等价关系是一种二元关系，满足自反，对称，传递。对于 TS 有很多种不同的等价定义方式

TS homomorphism

考虑两个 TS $A = (S, S_0, T, \alpha, \beta)$ 与 $A' = (S', S_0', T', \alpha', \beta')$

从 A 到 A' 的 homomorphism 是一对映射 (h_σ, h_τ)

$$h_\sigma : S \to S'$$
$$h_\tau : T \to T'$$

满足 $\forall t \in T$

$$\alpha'(h_\tau(t)) = h_\sigma(\alpha(t))$$
$$\beta'(h_\tau(t)) = h_\sigma(\beta(t))$$

对于 labeled TS 还需满足

$$\lambda'(h_\tau(t)) = \lambda(t)$$

如果 h_σ, h_τ 都是 surjective，则称 homomorphism h 是 surjective，A' 是 A 在 h 下的 quotient

TS strong isomorphism

一个 TS strong isomorphism 是一个 TS homomorphism，满足 h_σ, h_τ 都是 **bijective**

显然如果 h 是 isomorphism 则 h 的逆也是一个 isomorphism

对于两个 strong isomorphic 的 TS，它们唯一的区别就是命名方式

TS weak isomorphism

对于一个 TS T，其可达的状态集合定义为

$$reach(T) = \{s : s_0 \twoheadrightarrow s\}$$

则如果两个 TS 的 **isomorphism** 是定义在 $reach(T)$ 上的，称这两个 TS weak isomorphism，即仅在可达的部分相同

显然，如果两个 TS isomorphic，则这两个 TS weak isomorphic

Bisimulation

令 T, T' 为两个 TS，则 bisimulation 是一个二元关系 $B \subseteq S \times S'$，其定义为

Basis. $B(s_0, s_0')$

Induction.

- 如果 $B(s_1, s_1')$ 且 $s_1 \to s_2$，则存在 $s_2' \in S'$ s. t. $s_1' \to s_2'$ 并且 $B(s_2, s_2')$
- 如果 $B(s_1, s_1')$ 且 $s_1' \to s_2'$，则存在 $s_2 \in S$ s. t. $s_1 \to s_2$ 并且 $B(s_2, s_2')$

> 这里的 transition 需要 label 相同（或是对应）

T, T' 是 bisimulation equivalence $\iff$ 存在 bisimulation

两个 isomorphic 的 TS 一定是 bisimilar 的，但反之不真

Free Product of TS

考虑 n 个 TS $A_i = (S_i, S_{0_i}, T_i, \alpha_i, \beta_i)$

则其 free product $A = A_1 \times A_2 \times \cdots \times A_n$ 定义为

$$
\begin{gathered}
A = (S, S_0, T, \alpha, \beta) \\
S = S_1 \times S_2 \times \cdots \times S_n \\
T = T_1 \times T_2 \times \cdots \times T_n \\
\alpha(t_1, t_2, \ldots, t_n) =< \alpha_1(t_1), \ldots, \alpha_n(t_n) > \\
\beta(t_1, t_2, \ldots, t_n) =< \beta_1(t_1), \ldots, \beta_n(t_n) >
\end{gathered}
$$

对于 labeled TS 同理

然而对于 product TS 来说，不是所有 transition 都是有意义的，因为要受到同步的限制

故有意义的 TS 是 free product 的一个子系统，其定义可表示为 **synchronous product**

如果 TS $A_1, A_2, \ldots, A_n$ 对应的 label alphabet 为 $\Sigma_1, \Sigma_2, \ldots, \Sigma_n$，且 $I \subset \Sigma_1 \times \Sigma_2 \times \cdots \times \Sigma_n$ 是一个同步限制条件，则其 synchronous product 记为

$$
< A_1, A_2, \ldots, A_n, I >
$$

是 free product 的子系统，仅包含满足 $\lambda(t) \in I$ 的 transition t

在 free product 中，假设所有的 TS 都是同步执行的，但是有时候也需要体现出某个 TS 执行而其他 TS 没有执行的情况，故可以引入 τ-transition，即从任意状态 s **到其自身**的 transition

在不同 TS 间有 shared label 的情况下 τ-transition 十分重要

Temporal Logic

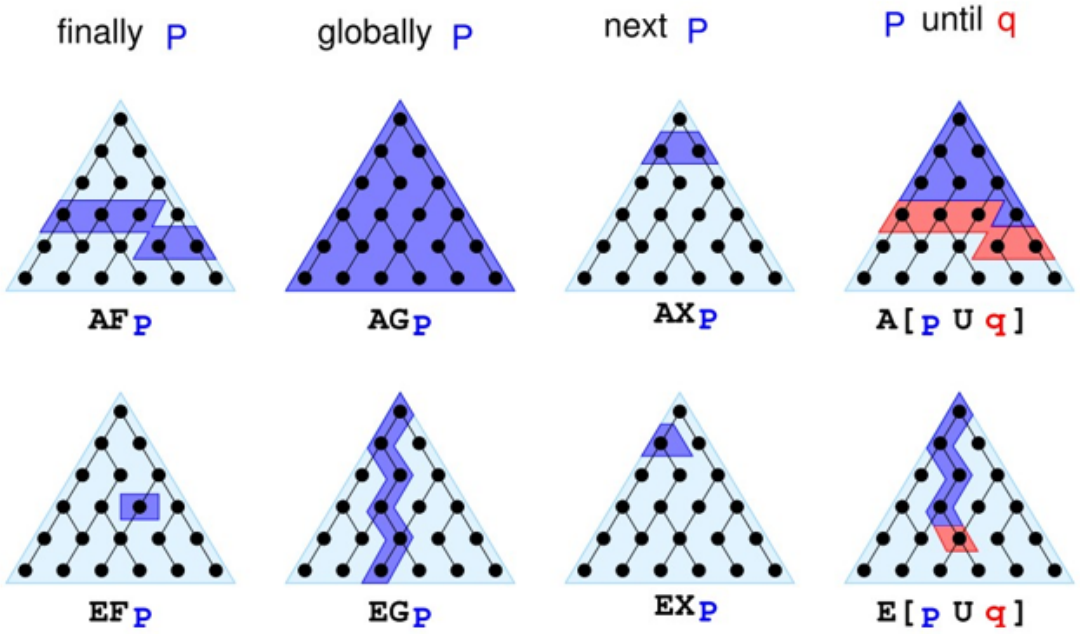
Temporal Logic 用于形式化描述 reactive system 中 transition 的序列，而其中的属性可以用 CTL* 来描述

CTL* 可以用来描述 computation tree 的属性，computation tree 是从 TS 开始状态的所有可能执行路径

CTL* 的组成包括

- path quantifier: A (for all computation path), E (for some computation path)
- temporal: X, F, G, U, R
  - X (next time): the property holds in the second state of the path
  - F (eventually): the property will hold at some state on the path
  - G (always): the property holds at every state on the path
  - U (until): if there is a state on the path where the second property holds, at every preceding state, the first one holds
  - R (release): the second property holds along the path up to and include the first state where the first property holds. However, the first property is not required to hold eventually

一个例子如下



常见的用 CFL 描述的行为有

- Safety: something bad will not happen，常用 AG 描述
- Liveness: something good will happen，常用 AF 描述
- Fairness: something is successful/allocated **infinitely often**，常用 AGAF 描述，即对于所有路径上的所有状态，都满足在之后的所有路径上都会在某个状态上满足条件