Transition system

(Redirected from State transition system)

In theoretical computer science, a **transition system** is a concept used in the study of computation. 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. If the label set is a singleton, the system is essentially unlabeled, and a simpler definition that omits the labels is possible.

Transition systems coincide mathematically with abstract rewriting systems (as explained further in this article) and directed graphs. They differ from finite-state automata in several ways:

- The set of states is not necessarily finite, or even countable.
- The set of transitions is not necessarily finite, or even countable.
- No "start" state or "final" states are given.

Transition systems can be represented as directed graphs.

Formal definition

Formally, a **transition system** is a pair (S, T) where S is a set of states and T, the *transition relation*, is a subset of $S \times S$. We say that there is a transition from state p to state q iff $(p, q) \in T$, and denote it $p \to q$.

A **labelled transition system** is a tuple (S, Λ, T) where S is a set of states, Λ is a set of labels, and T, the *labelled transition relation*, is a subset of $S \times \Lambda \times S$. We say that there is a transition from state p to state q with label α iff $(p, \alpha, q) \in T$ and denote it

$$p \xrightarrow{\alpha} q\,.$$

Labels can represent different things depending on the language of interest. Typical uses of labels include representing input expected, conditions that must be true to trigger the transition, or actions performed during the transition. Labelled transitions systems were originally introduced as *named* transition systems.[1]

Special cases

- If, for any given p and α, there exists only a single tuple (p, α, q) in T, then one says that α is *deterministic* (for p).

- If, for any given p and α, there exists at least one tuple (p, α, q) in T, then one says that α is *executable* (for p).

Coalgebra formulation

The formal definition can be rephrased as follows. Labelled state transition systems on S with labels from Λ correspond one-to-one with functions $S \to \mathcal{P}(\Lambda \times S)$, where $\mathcal{P}$ is the (covariant) powerset functor. Under this bijection (S, Λ, T) is sent to $\xi_T : S \to \mathcal{P}(\Lambda \times S)$, defined by

$$p \mapsto \{\, (\alpha, q) \in \Lambda \times S \mid p \xrightarrow{\alpha} q \,\}.$$

In other words, a labelled state transition system is a coalgebra for the functor $P(\Lambda \times -)$.

Relation between labelled and unlabelled transition system

There are many relations between these concepts. Some are simple, such as observing that a labelled transition system where the set of labels consists of only one element is equivalent to an unlabelled transition system. However, not all these relations are equally trivial.

Comparison with abstract rewriting systems

As a mathematical object, an unlabeled transition system is identical with an (unindexed) abstract rewriting system. If we consider the rewriting relation as an indexed set of relations, as some authors do, then a labeled transition system is equivalent to an abstract rewriting system with the indices being the labels. The focus of the study and the terminology are different, however. In a transition system one is interested in interpreting the labels as actions, whereas in an abstract rewriting system the focus is on how objects may be transformed (rewritten) into others.[2]

Extensions

In model checking, a transition system is sometimes defined to include an additional labeling function for the states as well, resulting in a notion that encompasses that of Kripke structure.[3]

Action languages are extensions of transition systems, adding a set of *fluents F*, a set of values *V*, and a function that maps $F \times S$ to V.[4]

See also

- Transition monoid
- Transformation monoid
- Semigroup action
- Simulation preorder
- Bisimulation
- Operational semantics

- Kripke structure
- Finite-state machine
- Modal μ-calculus

References

1. Robert M. Keller (July 1976) "Formal Verification of Parallel Programs (https://dl.acm.org/citation.cfm?id=360251)", *Communications of the ACM*, vol. **19**, nr. *7*, pp. 371–384.
2. Marc Bezem, J. W. Klop, Roel de Vrijer ("Terese"), *Term rewriting systems*, Cambridge University Press, 2003, ISBN 0-521-39115-6. pp. 7–8.
3. Christel Baier; Joost-Pieter Katoen (2008). *Principles of Model Checking*. The MIT Press. p. 20. ISBN 978-0-262-02649-9.
4. Micheal Gelfond, Vladimir Lifschitz (1998) "Action Languages", *Linköping Electronic Articles in Computer and Information Science*, vol. **3**, nr. *16*.