Principles of Abstract Interpretation
MIT press

Ch. **43**, Transition Semantics

Patrick Cousot

pcousot.github.io

PrAbsInt@gmail.com github.com/PrAbsInt/

These slides are available at
http://github.com/PrAbsInt/slides/slides-43--transition-semantics-PrAbsInt.pdf

Chapter 43

Ch. **43**, Transition Semantics

Introduction

© P. Cousot, NYU, CIMS, CS

Objectives

- A further abstraction of a eyeful trace semantics is into a transition system
- A relation between a state and its potential successor states
- This is a simple abstraction

Transition system

© P. Cousot, NYU, CIMS, CS

Transition system

- A transition system is a triple $\langle\Sigma,\ \mathbb{I},\ \xrightarrow{\tau}\rangle$ where Σ is a non-empty set of states σ, $\mathbb{I} \subseteq \Sigma$ is a set of initial states, and $\xrightarrow{\tau} \in \wp(\Sigma \times \Sigma)$ is a transition relation between a state and its possible successors.

© P. Cousot, NYU, CIMS, CS

Transition system abstraction

- A transition system $\langle \Sigma,\ \mathbb{I},\ \xrightarrow{\tau} \rangle$ can be used to define a state prefix trace semantics as follows.

$$\gamma^\tau(\langle \Sigma,\ \mathbb{I},\ \xrightarrow{\tau} \rangle) \quad \triangleq \quad \{\pi_0 \cdot \cdots \cdot \pi_n \mid n \in \mathbb{N} \wedge \pi_0 \in \mathbb{I} \wedge \forall i \in [0, n[\ .\ \pi_i \xrightarrow{\tau} \pi_{i+1}\} \quad (43.1)$$

(where $\sigma \xrightarrow{\tau} \sigma'$ is a shorthand for $\langle \sigma,\ \sigma' \rangle \in \xrightarrow{\tau}$.)

- Conversely a prefix trace semantics S can be abstracted in a transition system

$$\alpha^\tau(S) \quad \triangleq \quad \langle \Sigma,\ \mathbb{I},\ \xrightarrow{\tau} \rangle \qquad (43.2)$$

where

$$\begin{array}{rcl} \Sigma & \triangleq & \{\pi_i \mid \exists n \in \mathbb{N}, \pi_0, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_n \ .\ \pi_0 \cdots \pi_n \in S\} \qquad (\text{or } \ \mathbb{S}) \\ \mathbb{I} & \triangleq & \{\pi_0 \mid \exists n \in \mathbb{N}, \pi_1, \ldots, \pi_n \ .\ \pi_0 \cdots \pi_n \in S\} \\ \xrightarrow{\tau} & \triangleq & \{\pi_i \longrightarrow \pi_{i+1} \mid \exists n \in \mathbb{N}^+, \pi_0, \ldots, \pi_{i-1}, \pi_{i+2}, \ldots, \pi_n \ .\ \pi_0 \cdots \pi_n \in S\} \end{array}$$

- This is a Galois connection

$$\langle \wp(\mathbb{T}^+),\ \subseteq \rangle \xleftrightarrows[\alpha^\tau]{\gamma^\tau} \langle \{\langle \Sigma,\ \mathbb{I},\ \xrightarrow{\tau} \rangle \mid \Sigma \in \wp(\mathbb{S}) \wedge \mathbb{I} \subseteq \Sigma \wedge \xrightarrow{\tau} \subseteq \Sigma \times \Sigma\},\ \dot{\subseteq} \rangle$$

© P. Cousot, NYU, CIMS, CS

Transition system abstraction (cont'd)

- In general information is lost by the abstraction of a prefix trace semantics to a transition system (take for example $\Pi = \{a, aa\}$ so that $\gamma^\tau \circ \alpha^\tau(\Pi) = a^+$ is the set fo all non-empty finite sequences of "a"s).
- Notice that the abstraction of the prefix trace semantics of a program into a transition system will only comprehend reachable states.
- So the transition semantics for a language is the join of all transition systems of the prefix trace semantics of all programs in the semantics.
- This may still be a strict overapproximation.

Transition semantics

- The transition semantics of the programming language $\mathbb{P}$ of Chapters **4** and **6** is

$$\alpha^{\tau}(\widehat{\mathcal{S}}^{*}_{\mathbb{S}}[\![\mathsf{S}]\!]) \quad = \quad \langle \mathbb{S},\ \{\langle \mathsf{at}[\![\mathsf{S}]\!],\ \rho\rangle \mid \mathsf{S} \in \mathbb{Pc} \wedge \rho \in \mathbb{Ev}\},\ \widehat{\mathcal{S}}^{\tau}[\![\mathsf{S}]\!]\rangle$$

- $\widehat{\mathcal{S}}^{\tau}[\![\mathsf{S}]\!]$ is defined by structural induction on program components $\mathsf{S} \in \mathbb{Pc}$ as follows.

 © P. Cousot, NYU, CIMS, CS

Transition semantics (cont'd)

Transition semantics of an assignment statement $\texttt{S} ::= {}^{\ell}\ \texttt{x = A ;}$

$$\widehat{\mathscr{S}}^{\,\tau}[\![\texttt{S}]\!] \quad = \quad \{\langle \ell,\ \rho\rangle \longrightarrow \langle \mathsf{after}[\![\texttt{S}]\!],\ \rho[\texttt{x} \leftarrow \mathscr{A}[\![\texttt{A}]\!]\rho]\rangle \mid \rho \in \mathbb{E}\mathbb{v}\} \tag{43.4}$$

Transition semantics of a statement list $\texttt{Sl} ::= \texttt{Sl}'\ \texttt{S}$

$$\widehat{\mathscr{S}}^{\,\tau}[\![\texttt{Sl}]\!] \quad = \quad \widehat{\mathscr{S}}^{\,\tau}[\![\texttt{Sl}']\!] \cup \widehat{\mathscr{S}}^{\,\tau}[\![\texttt{S}]\!] \tag{43.5}$$

Transition semantics of an iteration statement $\texttt{S} ::= \texttt{while}\ {}^{\ell}\ \texttt{(B)}\ \texttt{S}_b$

$$\begin{aligned}\widehat{\mathscr{S}}^{\,\tau}[\![\texttt{while}\ {}^{\ell}\ \texttt{(B)}\ \texttt{S}_b]\!] \quad = \quad & \big\{\langle \ell,\ \rho\rangle \longrightarrow \langle \mathsf{after}[\![\texttt{S}]\!],\ \rho\rangle \mid \mathscr{B}[\![\texttt{B}]\!]\,\rho = \mathsf{ff}\big\} \\ & \cup \big\{\langle \ell,\ \rho\rangle \longrightarrow \langle \mathsf{at}[\![\texttt{S}_b]\!],\ \rho\rangle \mid \mathscr{B}[\![\texttt{B}]\!]\,\rho = \mathsf{tt}\big\} \cup \widehat{\mathscr{S}}^{\,\tau}[\![\texttt{S}_b]\!]\end{aligned} \tag{43.6}$$

© P. Cousot, NYU, CIMS, CS

Transition semantics (cont'd)

Transition semantics of a break statement $\mathtt{S} ::= {}^{\ell}\ \mathtt{break}\ \mathtt{;}$

$$\widehat{\mathcal{S}}^{\tau}[\![\mathtt{break}\ \mathtt{;}]\!] = \{\langle \ell,\ \rho\rangle \longrightarrow \langle \mathsf{break\text{-}to}[\![\mathtt{S}]\!],\ \rho\rangle \mid \rho \in \mathbb{E}\mathrm{v}\} \tag{43.7}$$

Theorem 43.11 The stateful prefix trace semantics $\mathcal{S}^{*}_{\mathbb{S}}$ of Section **42.2** for the programming language $\mathbb{P}$ of Chapter **4** is generated by the above transition semantics (of Chapter **43**).

© P. Cousot, NYU, CIMS, CS

Conclusion

Conclusion I

- We made the link between the stateful prefix trace semantics of Chapter **42** and the more traditional stateful small step operational semantics where execution traces or reachable states are defined by a transition semantics *e.g*. [Wegner, 1972a] stating "Each of the formalisms above involves the notion of a *state set* consisting of the set of all information configurations that can occur during computations and a *state transition relation* which defines for each state a next state or set of next states.".
- The transition semantics can be implemented in the language it defines, which is John Reynolds' idea of definitional interpreters of [Reynolds, 1998] (following the definition of Lisp in Lisp [McCarthy, Abrahams, Edwards, Hart, and Levin, 1966, APPENDIX B, p. 70–72]).
- [Moggi, Tahab, and Thunberg, 2020] introduces monadic transition systems as a generalization of transition systems which unifies a wide range of models, including deterministic automata, non-deterministic automata, Markov chains, and probabilistic automata.

Conclusion II

- The operational semantics dates back to John McCarthy [McCarthy, 1960]. The Vienna Definition Language (VDL) [Jones, 1978; Wegner, 1972b] was a formal specification language essentially used for giving operational semantics descriptions [Henhapl and Jones, 1978].
- The aura of operational semantics faded away with the emergence of Dana Scott and Christopher Strachey's denotational semantics [Milne and Strachey, 1976; Scott and Strachey, 1971; Stoy, 1981; Tennent, 1981], most people thinking that describing what is done instead of how it is done is more concise and elegant.
- Denotational semantics introduced the idea of structural definition and the use of fixpoints to handle iteration and recursion.
- It reached its limits when dealing with parallelism, for which no viable denotational solution ever emerged.

 © P. Cousot, NYU, CIMS, CS

Conclusion III

- The come back of operational semantics is due to Gordon Plotkin [Plotkin, 2004] thanks to the use of structural rule-based deductive definitions (Chapter **16**), considered a gracefulness of style by many.
- Moreover, it applies to parallelism by interleaving of atomic actions.
- However, with weak memory models of modern machines [Alglave, 2012], the use of states is somewhat heavy so stateless models might be an interesting alternative [Alglave and Cousot, 2016], hardly describable elegantly and concisely by state-based transition systems.
- This motivates our choice of starting from a stateless prefix trace semantics in Chapter **6**, instead of the more traditional stateful semantics of Chapter **42**, that has been easily recovered from Chapter **6** by abstract interpretation.

 © P. Cousot, NYU, CIMS, CS

Bibliography

References I

Alglave, Jade (2012). "A formal hierarchy of weak memory models". *Formal Methods in System Design* 41.2, pp. 178–210.

Alglave, Jade and Patrick Cousot (2016). "Syntax and analytic semantics of LISA". *CoRR* abs/1608.06583.

Henhapl, Wolfgang and Cliff B. Jones (1978). "A Formal Definition of Algol 60 as Described in the 1975 Modified Report". In: *The Vienna Development Method: The Meta-Language*. Vol. 61. Lecture Notes in Computer Science. Springer, pp. 305–336.

Jones, Cliff B. (1978). "The META-Language: A Reference Manual". In: *The Vienna Development Method: The Meta-Language*. Vol. 61. Lecture Notes in Computer Science. Springer, pp. 218–277.

McCarthy, John (1960). "Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I". *Commun. ACM* 3.4, pp. 184–195.

References II

McCarthy, John, Paul W. Abrahams, Daniel J. Edwards, Timothy P. Hart, and Michael I. Levin (1966). *LISP 1.5 Programmer's Manual*. MIT Press.

Milne, Robert and Christopher Strachey (1976). *A Theory of Programming Language Semantics (2 Vol)*. Springer.

Moggi, Eugenio, Walid Tahab, and Johan Thunberg (2020). "Sound Over-Approximation of Probabilities". *Acta Cybernetica*. to appear.

Plotkin, Gordon D. (2004). "A structural approach to operational semantics". *J. Log. Algebr. Program.* 60-61, pp. 17–139.

Reynolds, John C. (1998). "Definitional Interpreters Revisited". *Higher-Order and Symbolic Computation* 11.4, pp. 355–361.

Scott, Dana S. and Christopher Strachey (Aug. 1971). *Towards a Mathematical Semantics for Computer Languages*. Technical report PRG-6. Oxford University Computer Laboratory, p. 49.

References III

Stoy, Joseph E. (1981). *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory*. Computer Science Series. MIT Press.

Tennent, Robert D. (1981). *Semantics of Programming Languages*. Prentice Hall.

Wegner, Peter (Jan. 1972a). "Operational Semantics of Programming Languages". *SIGPLAN Not.* 7.1, pp. 128–141.

— (1972b). "The Vienna Definition Language". *ACM Comput. Surv.* 4.1, pp. 5–63.

Home work

Read Ch. **43** "Transition Semantics" of

Principles of Abstract Interpretation
Patrick Cousot
MIT Press

The End, Thank you

 © P. Cousot, NYU, CIMS, CS