Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

Verification of Partial Properties
Introduction

Section 3.1 of "Correctness Preserving Transformation"
ESPRIT Project 2304 LOTOSPHERE

A. Fantechi

Nota Interna B4-35
Agosto 1990

Section 3.1

Introduction

3.1.1 The meaning of "Partial Properties"

By "partial property" we mean a partial representation of the behaviour of a system, which abstracts from details which are considered as irrelevant for that representation. The main purpose of this chapter is to define notions of correctness given by partial properties, in order to complement the usual notion of correctness as an equivalence of two complete specifications, studying in detail the possible roles, advantages and application techniques of partial properties in the context of correctness preserving transformations. This concept is opposed to that of a "total" representation of the behaviour of the system, such as that considered, for instance, when we want to verify the equivalence of a given specification and its implementation. In this sense, the other chapters of this document on Correctness Preserving Transformations are mostly concerned with the preservation of "total" properties.

Simple classes of partial properties

As simple instances of partial properties, we can first mention the structural properties of the specification, like the style in which the specification is written (constraint-oriented, state-oriented,...), or particular views on the specification, intended as restrictions of its sort (the set of its visible gates).

It is however more interesting to consider semantic properties, which can be imposed as requirements on the specification. We may, for instance, refer to the classical distinction of partial properties into *safety* (something bad never happens) and *liveness* properties (something good eventually happens) [L83] or to the finer classification given in [MP89], in which liveness properties are furtherly divided in finer classes, like *termination* (a good thing happens in some state of the computation), *persistence* (something good happens in all, except for finitely many, states of the computation), *recurrence* (something good happens in infinitely many states of the computation), or *response* (the occurrence of an event always triggers another event). We can also identify other, generally useful, classes of properties such as *potentiality* (there is the possibility that something will happen).

Some examples of partial properties

Let us consider the following natural language specification of the properties of a database system:
1) A user cannot access the database if he/she has not correctly typed his/her password.
2) Whenever a logged user queries the database, he/she eventually gets a reply from it.
3) The database system may refuse the service to a user (e.g. due to its own capacity limits).
We can recognize the first point as a safety property, the second as a response property (also a liveness property), and the third as a potentiality property.

Other useful properties can often be considered as belonging to one of the above classes. For example the insensitiveness of a specification to the underlying form of communication (synchronous or asynchronous) can be seen as a response property: every sent message will eventually arrive at destination.

3.1.2. Formalisms to express partial properties

Temporal Logic

Temporal Logic [MP81, P81, L83] has been long recognized as a valid tool to specify and verify properties of concurrent systems, working at a higher level of abstraction with respect to the constructive formalisms used for specifying concurrent systems such as LOTOS.

Formulae in temporal logic, do not only express the truth of a certain statement, but its truth relative to a point in time, using connectives such as "next", "always", "eventually", "until", and so on. In this way, we can express statements of systems which evolve in time.

For example, the statement seen above: "Whenever a logged user queries the database, he/she eventually gets a reply from it", can be expressed by the temporal logic formula:

always (query implies eventually (reply))

where query and reply are predicates which are true if the events of querying the database or of getting a reply from it, occur.

An important distinction of the possible temporal logic definitions is made on the basis of the structure of time chosen. If the structure of time considered is linear (each moment has exactly one future) we have a "linear temporal logic". If the structure of time is "branching" (every moment may have more than one future) we have a "branching temporal logic". This distinction is important for the expression of properties: safety and liveness properties are expressed in linear temporal logic, but potentiality properties can be expressed only by branching temporal logic.

The question of the expressibility of fairness properties in branching temporal logic has, in the past, given rise to considerable discussion. [L81, EH86].

Partial Process Specifications

Another way to express partial properties is by partial process specifications, in a language which can be either LOTOS itself or an extended "annotated" version of it. One must provide in addition *procedures of reductions* to be applied to the (concrete) system under analysis, before it is compared with the partial specification. The idea here is twofold: first, only certain aspects

of the behaviours are focussed on, discarding others, and thus the process, seen from a certain angle, is mapped to the desired property expressed as another process; second, complementary pertinent items of information, increasing practical expressivity, can be figured explicitly in an appropriate form both in the specification and in the process reduction, thus providing further insights.

As an instance of the first case, the theory of "process abstraction" was devised in [B85]. A syntax allowing the definition of "abstract actions", as in databases transactions for instance, is introduced in that paper. Following a linear time philosophy, a search is then made for paths of behaviours in the process; the result is an "abstract transition system", usually much simpler to comprehend. We shall call *experiments* these sequences of behaviours which belong to a given abstract action. A fairly common use of abstract actions is refutation: a *Positive* set of experiments is introduced, which intuitively should lead to well-identified configurations of the system. They are supplemented by another set *Negative* - those which should not be possible. The presence of the second type of abstract actions in the reduced system is thus highly indicative, since they identify locations (or configurations) where faulty behaviours may occur. This particular verification could be summed up in logical terms:

always (no experiment from Negative after an experiment from Positive),

and bears strong links with the theory of testing. This approach was exemplified in [BRSV89].

As an instance of the second case, divergence predicates can carry information to the user on invisible aspects of the processes, such as livelock states.

Connections between temporal logics and reduction can be drawned: Propositional Dynamic Logic [P79] uses abstract actions as parameters to the logical modalities, so that [R]f means "always f after any experiment belonging to R". Proving such a property can be reduced to proving a simpler one, (in the typical form of Hennessy-Milner logic [HM85]), on the abstract reduced system. The relevant modalities may often be interpreted on the reduced systems, as formulae to check are quite regular. In any case, introduction of a restricted version of modalities in the shape of the partial specifications to check may be obtained using so-called modal specifications [LT88].

We can note that also the expression of properties as constraints on the specification in a constraint oriented style [B89] can be considered as an instance of the use of partial process specification.

3.1.3 Use of partial properties

Verification and analysis

Partial properties can be used as "commitments" for the implementation, serving as guidelines for the transformation from the specification to an implementation. It should be verified that each refinement step satisfies the commitments. On the other hand, the partial properties satisfied by a LOTOS specification can be derived from the specification itself. Derived properties provide a different view, usually more abstract and less implementation oriented, of the specified system; they can be used to produce assertions that provide a more readable view for a user who is not concerned about architectural details.

A particular case of specification analysis is debugging: deriving a partial view (a "reduction") from a LOTOS specification is promising as a way of checking partial correctness and actually

debugging a LOTOS program under development. Instead of formalising an assertion on the system, it can be derived directly. The synthesized property may sometimes be different from that expected; such a case will suggest misconceptions in the process under analysis, and is a starting point for a diagnostics. This is after all what partial verification should provide, in the absence of total verifications insuring error-free program derivations.

Moreover, we should note that there are classes of typical properties that can be requested of a system and which cannot be expressed directly in LOTOS terms: fairness, divergence, or dually the computation progress (the necessary occurrence of events). Two possible ways of relating these properties to LOTOS specifications are the following:
1) Assertions can be used to give additional constraints to the process structure, in order to express properties of this class, so that existing specifications are "enriched" by additional property expressions; this case may also be seen as an extension of the specification language
2) It is possible to verify that, under certain constraints, a given LOTOS specification will satisfy given properties which are not expressible in LOTOS terms: for instance, we can verify that, when assuming some general fairness constraints which are not expressed in the specification, a particular response property is satisfied.

Techniques to achieve verification/derivation of partial properties.

Modal Characterization [HM85] and Temporal Semantics [P81, B87a] provide ways to deduce a logic characterization of the properties enjoyed by a system starting from its LOTOS specification. Once a logic formula representing the behaviour of the system has been obtained, the proof that the system satisfies a given property is reduced to verifying that the formula implies the logic expression of the property itself. Logical deduction can therefore be used both to derive and to verify properties of a given specification.

The equivalent to logical deduction when properties are expressed by process expressions is abstraction: we can derive a partial view of a LOTOS specification by basing the various ways to perform reductions (which could be called transition system "morphisms") on the syntactic notions of abstractions, in which new "abstract" actions are devised, each as a language of sequences of more concrete actions. This can also be viewed also as a way of specifying (and not *programming*) some levels of granularity with respect to the atomicity of sequences of actions. Abstraction can be seen therefore as a transformation changing the atomicity level; the notion of correctness preserved by this transformation is a bisimulation relation, parameterised by the set of actions from which one wants to abstract.

The model checking approach considers transition systems as suitable models of a temporal logic; thus, verifying that a LOTOS specification P satisfies a property expressed by the temporal logic formula F amounts to checking that the transition system associated by the operational semantics to P is a model for F. This approach, which is suited for property verification, can be supported by automatic tools which perform checking on finite transition systems [CES86].

3.1.4 The scope of the present chapter

The explicit, formal definition of the partial properties of LOTOS specifications represents a powerful tool for their analysis. For example, partial properties can be useful to compare LOTOS specifications using non-constructive relations (comparing two specifications by comparing their set of partial properties). In this sense, they are extremely useful to formally

define the "correctness preserving relation" R_{CP} used in the context of correctness preserving transformation as shown in chapter 1. It should be noted that, since partial properties are usually given at a higher abstraction level than LOTOS specifications, this principle extends to the case of "non LOTOS to LOTOS" and "LOTOS to non LOTOS" transformations, where the lack of any equivalence concepts inhibits any formal reasoning on correctness preservation.

An important subcase of transformations in which partial properties can play an important role is the case of style transformation. Particular properties may be associated with specific styles, while interesting properties to be preserved in the transformation should be independent of the style.

We want to mention the relations of partial properties with formal methods for testing, studied within the Task 1.3: in fact, the acceptance or rejection of a test is a partial property. The expression of tests can be seen as related to the description of partial properties by process expressions.

Three different approaches, which cover the theme of partial properties, are presented in this chapter, each in one of the next three sections.

The first approach aims to relate the LOTOS world to the temporal logic world, defining a Temporal Semantics for Basic LOTOS, that is, a function defined compositionally on the Basic LOTOS terms, which gives the temporal logic formula associated to a LOTOS term (Section 3.2). Some verification examples achieved by relating the Temporal Semantics of some processes to formulae expressing desired properties are shown. Anyway, this approach is particularly promising as an analysis tool of the LOTOS language itself.

The second approach aims to pursue the verification of partial properties by first performing a reduction (imposing a "view" of the system at hand) and then reasoning with temporal logic model checking (Section 3.3). The approach is mainly aimed at obtaining an "explanatory model" of the transition system (associated with a LOTOS specification) with respect to a set of partial properties expressed in temporal logic: the explanatory model and the LOTOS specification being semantically compatible with respect to the set of partial properties which has been selected. The compound method is based upon an extensive use of the standard observational equivalence. A new equivalence, the so-called behaviour equivalence, is introduced; its aim is to deal symmetrically with states and events and to take into account livelock problems. Expressivity associated with behaviour equivalence is comparable to that associated with temporal logic CTL (Computation Tree Logic) [CE81, S82].

The third approach is a more general framework for reductions of processes through transition systems, showing how particular syntactic means to set abstraction on behaviours can produce powerful reductions (Section 3.4). In this approach the accent is posed on the use of reduction principles for debugging LOTOS specifications.

The three approaches employ different constraints on the LOTOS specifications that can be handled, in order to effectively achieve early meaningful results: Temporal Semantics is given for Basic LOTOS, in order to avoid the complexities of the treatment of values, and in the present version is given consistently with "strong" trace semantics (unobservable actions are treated not differently from observable ones); the other approaches, dealing mainly with transition systems, have less concerns about the language itself, but are forced to consider finite state systems.

Such constraint allow however automatic support of the proposed approaches to be feasible. Even if complete automatic support is out of the scope of this document, some considerations on potential tool support are given in Section 3.5.

References

[A87] S. Abramsky, "Observation Equivalence as Testing equivalence", *Theoretical Computer Science*, Vol.53, 1987.

[B85] G. Boudol, Notes on Algebraic Calculi of Processes", in K. Apt ed., "Logics and Models of Concurrent Systems", Nato ASI Series F13, 1985, pp 261-303.

[B87a] H. Barringer, "The Use of Temporal Logic in the Compositional Specifications of Concurrent Systems", in "Temporal Logics and their applications", A. Galton, ed., Academic Press, London, 1987, pp.53-90.

[B87b] G. Boudol, "Communication is an abstraction", INRIA, RR636, March 1987.

[B89] E. Brinskma, "Constraint-oriented specification in a constructive formal description technique" in "Stepwise Refinement of Distributed Systems", Proc. REX Workshop, Mook, 29 May - 2 June 1989, W.P. de Roever (ed.), *Lecture Notes in Computer Science*, Springer Verlag.

[BB87] T. Bolognesi, E. Brinskma, "Introduction to the ISO Specification Language LOTOS", *Computer Networks & ISDN Systems*, vol. 14, n. 1, January 1987, pp. 25-29.

[BB89] B. Banieqbal, H. Barringer: "Temporal Logic Fixed Point Calculus", Proceedings Colloquium on Temporal Logic and Specification, Altrincham, England, 1987. *Lecture Notes in Computer Science*, vol.398, 1989, pp.62-74.

[BC88] T. Bolognesi, M. Caneve, "Squiggles - a Tool for the Analysis of LOTOS Specifications", Proceedings 1st International Conference on Formal Description Techniques (FORTE'88), K.J. Turner, ed., North-Holland, 1989, pp. 201-216.

[BGS88] A. Bouajjani, S. Graf, J. Sifakis, "A Logic for the Description of Behaviours and Properties of Concurrent Sistems", REX School /Workshop, *Lecture Notes in Computer Science* vol. 354, May 1988, pp.398-410.

[BKP84] H. Barringer, R. Kuiper, A.Pnueli: "Now you may Compose Temporal Logic Specifications", Proc. 16th ACM Symposium on the Theory of Computing, 1984, pp. 51-63.

[BKP85] H. Barringer, R. Kuiper, A.Pnueli: "A Compositional Temporal Approach to a CSP-like Language", in E.J. Neuhold and G. Chroust eds., Formal Models of Programming, IFIP, North-Holland, pp. 207-227.

[BL 90] G. Boudol, K. Larsen, "Graphical vs. Logical Specifications", to appear in Proceedings CAAP '90, Copenhagen, May 1990.

[BRSV89] G. Boudol, V. Roy, R. de Simone, D. Vergamini, "Process Calculi, from Theory to Practice: Verification Tools", in Automatic Verification Methods for Finite State Systems, *Lecture Notes in Computer Science*, vol.407, pp.1-10.

[CE81] E.M Clarke, E.A Emerson,"Synthesis of synchronization skeletons for branching time temporal logic", Proceedings of the Workshop on Logic Programs, Yorktown Heights New York, *Lectures Notes in Computer Science*, Volume 131, 1981, pp. 52-71.

[CES86] E. M. Clarke, E.A. Emerson, A.P. Sistla, "Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specification", *ACM Transactions on Programming Languages and Systems*, vol. 8, n. 2, pp. 244-263, 1986.

[CPS89] R. Cleaveland, J. Parrow, B. Steffen, "A Semantics Based Verification Tool for Finite State Systems", Ninth IFIP Symposium "Protocol Specification, Testing and Verification", Enschede (1989), North Holland.

[D86] A. Dicky, "An algebraic and algorithmic method for analysing transitions systems, *Theoretical Computer Science*, Vol. 46, n.2-3, 1986, pp. 285-303.

[D87] R. De Nicola, "Extensional Equivalences for Transition Systems", *Acta Informatica*, 24, pp. 211-237.

[D89] E. Dubuis, "An algorithm for translating LOTOS Behavior expressions into Automata and Ports", Proceedings 2nd International Conference on Formal Description Techniques (FORTE'89), Vancouver, Dec. 1989, pp. 215-229.

[dSV89] R. de Simone, D. Vergamini, "Aboard Auto", INRIA Technical Report 11, 1989

[EH86] E.A. Emerson, J.Y. Halpern, ""Sometimes' and 'Not Never' Revisited: On Branching versus Linear Time Temporal Logic", *Journal of the ACM*, vol. 33, n. 1, Jan. 1986, pp. 151-178.

[ES89] E.A Emerson, J. Srinivasan, "Branching Time Temporal Logic", in J.W. de Bakker, W.P. de Roever, G. Rozenberg, eds., "Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency", *Lecture Notes in Computer Science*, vol. 354, 1989, pp. 123-172.

[FGL88] A. Fantechi, S. Gnesi, C. Laneve, "A Proof of Simple Abstractness for the Temporal Semantics of LOTOS", I.E.I. Internal Report B4-38, August 1988.

[FGL89] A. Fantechi, S. Gnesi, C. Laneve, "An Expressive Temporal Logic for Basic LOTOS", Proceedings 2nd International Conference on Formal Description Techniques (FORTE'89), Vancouver, Dec. 1989, pp. 383-399.

[FLS89] M. Faci, L. Logrippo, B. Stépien, "Formal Specification of Telephone Systems in LOTOS", in E. Brinskma, G. Scollo, C.A. Vissers eds., Proc. of 9th IFIP WG 6.1 International Symposium on Protocol Specification, Testing and Verification, North-Holland 1989.

[H81] C.A.R. Hoare, "A Model for Communicating Sequential Processes", Technical Monograph Prg-22, Computing Laboratory, University of Oxford, 1981.

[HM85] M. Hennessy, R. Milner, "Algebraic Laws for Nondeterminism and Concurrency", *Journal of ACM*, vol. 32, n. 1, January 1985, pp. 137-161.

[K63] S.A Kripke, "Semantical Analysis of Modal Logic, *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, Vol. 9, 1963.

[L81] L. Lamport, "'Sometime' is Sometimes 'Not Never'", Proceedings of 7th Annual Symp. on Principles of Programming Languages, ACM, 1980, North Holland, 1983, pp.174-185.

[L83] L. Lamport, "What Good is Temporal Logic?", Proceedings of IFIP'83, North Holland, 1983, pp.657-668.

[L85] K. Larsen, "A Context Dependent Equivalence between Processes", Proc. ICALP'85, *Lecture Notes in Computer Science*, vol. 194, pp. 373-382.

[L90] K. Larsen, "Compositional Theories Based on an Operational Semantics of Contexts" in "Stepwise Refinement of Distributed Systems", Proc. REX Workshop, Mook, 29 May - 2 June 1989, W.P. de Roever (ed.), *Lecture Notes in Computer Science*, Springer Verlag.

[LT88] K. Larsen, B. Thompson, "A Modal Process Logic", Proc. LICS'88, pp. 203-210.

[M80] R. Milner, "A Calculus of Communicating Systems", *Lecture Notes in Computer Science* vol. 92, 1980.

[M81] R. Milner, "A modal characterization of observable machine-behaviour", Proceedings CAAP'81, *Lectures Notes in Computer Science*, Vol. 112, 1981, pp.25-34.

[MP81] Z. Manna, A. Pnueli, "Verification of Concurrent Programs: The Temporal Framework", in R.S. Boyer, J.S. Moore, eds., "Correctness Problem in Computer Science", Academic Press, 1981, pp.215-273.

[MP89] Z. Manna, A. Pnueli, "The Anchored Version of the Temporal Framework", in J.W. de Bakker, W.P. de Roever, G. Rozenberg, eds., "Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency", *Lecture Notes in Computer Science*, vol. 354, 1989, pp.201-284.

[MV89] E. Madeleine, D. Vergamini, "AUTO: A Verification Tool for Distributed Systems Using Reduction of Finite Automata Networks", Proceedings 2nd International Conference on Formal Description Techniques (FORTE'89), Vancouver, Dec. 1989, pp. 77-83.

[P79] R. Parikh, "Propositional Dynamic Logics of Programs: A survey" in "Logics of Programs", *Lecture Notes in Computer Science*, vol. 125, 1979, pp.102-144.

[P81] A. Pnueli, "The Temporal Semantics of Concurrent Programs", *Theoretical Computer Science*, vol.13, 1981, pp.45-60.

[P85] A. Pnueli, "Linear and Branching Structures in the Semantics and Logic of Reactive Systems" Proceedings of 12th ICALP, *Lecture Notes in Computer Science* vol. 194, July 1985, pp.15-32.

[P86] A. Pnueli : "Specification and Development of Reactive Systems", Proceedings IFIP Congress, North-Holland, 1986, pp. 845-858.

[P88] A. Pnueli, "Applications of Temporal Logic to the Specification and Verification of Reactive Systems: A Survey of Current Trends", REX School /Workshop, Noordwijkerhout, May 1988.

[R86] R. Rosner, "A Choppy Logic", Master Degree Thesis, Weizmann Institute of Science, January 1986.

[S82] J.Sifakis : "A unified approach for studying the properties of transition systems", *Theoretical Computer Science*, Vol 18, 1982, pp.227-258.

[S83] J.P Schwartz, "Quasar: une réalisation du système CESAR: Description, Spécification et Analyse des applications réparties, Thèse de Docteur Ingénieur 1983, Université Scientifique et Médicale de Grenoble.

[VGG89] R. J. Van Glabbeek, U. Goltz, "Equivalence Notions for Concurrent Systems and Refinement of Actions" MFCS 89, *Lecture Notes in Computer Science*, vol.379, pp. 237-248.

[WC89] J.P.Wu, S.T. Chanson, "Translation from LOTOS and Estelle Specifications to Extended Transitin Systems and its Verification", Proceedings 2nd International Conference on Formal Description Techniques (FORTE'89), Vancouver, Dec. 1989, pp. 677-697.