DOWNLOAD PDF FIRST-ORDER DYNAMIC LOGIC

Chapter 1 : CiteSeerX â€” Citation Query First Order Dynamic Logic

An abstract is not available. Koen V. Hindriks, Frank S. de Boer, Wiebe van der Hoek, John-Jules Ch. Meyer, A Programming Logic for Part of the Agent Language 3APL, Proceedings of the First International Workshop on Formal Approaches to Agent-Based Systems-Revised Papers, p, April

Show Context Citation Context Subordinate and reduction policies make e Security-sensitive environments protect their information resources against unauthorized use by enforcing access control mechanisms driven by access control policies. Due to the need to compare, contrast, and compose such protected information resources, access control policies regulating their mani Due to the need to compare, contrast, and compose such protected information resources, access control policies regulating their manipulation need to be compared, contrasted, and composed. An algebra for manipulating such access control policies at a higher propositional level, where the operations of the algebra are abstracted from their specification details, is the subject of this paper. This algebra is applicable to policies that have controlled nondeterminism and all or nothing assignments of access privileges in their specification. These requirements reflect current practices in discretionary and role-based access control models. Therefore, the proposed algebra can be used to reason about role-based access control policies combined with other forms of discretionary policies. We show how to use algebraic identities to reason about consistency, completeness, and determinacy of composed policies using similar properties of their constituents. On Being Responsible by N. Jennings , " A comprehensive theory describing A comprehensive theory describing this class of social interaction would need to cover at least the following aspects: The framework described herein defines the prerequisites for such action and also prescribes how agents should behave both in their own problem solving and with respect to other group members once the problem solving has been established. Typically in a community of autonomous agents, one of the primary motives for joint action is when no individual is capable of achieving a desired objective alone; only by combining and coordinating with others can the target be reached. Joint action is usually a reciprocal process in which participating agents augment their objectives and problem solving to comply with those of others - hence it is a fairly sophisticated form of cooperation. It requires greater knowledge, awareness and reflection by an agent both with respect to its own problem solving objectives and about their compatibility with the objectives of others, than simpler forms of social interaction such as task and result sharing [19]. Joint action, by definition, requires an objective the group wishes to achieve - it is the glue which binds the team together. As a consequence of the autonomous nature of the agents, team members will only participate if they can derive some benefit from This paper presents a particular approach to the design and verification of large sequential systems. It is based on structured algebraic specifications and stepwise refinement by program modules. The approach is implemented in Kiv Karlsruhe Interactive Verifier , and supports the entire desig The approach is implemented in Kiv Karlsruhe Interactive Verifier , and supports the entire design process starting from formal specifications and ending with verified code. Its main characteristics are a strict decompositional design discipline for modular systems, a powerful proof component, and an evolutionary verification model supporting incremental error correction and verification. We present the design methodology for modular systems, a feasible verification method for single modules, and an evolutionary verification technique based on reuse of proofs. We report on the current performance of the system, compare it to others in the field, and discuss future perspectives. There is currently a broad interest in dialogue acts and dialogue act taxonomies, and new uses, taxonomies, and standardization efforts continue to be proposed. This paper presents a discussion of issues that must be addressed in order to facilitate the shared understanding and use of taxonomies. The discussion is framed in terms of 20 questions, the answers to which will help make the meanings of taxonomy elements more clear to different communities of users.

Chapter 2 : First-order dynamic logic - David Harel - Google Books

Particularly, it only allows one to reason about partial correctness. That is, a partially correct program may have non-terminating executions. In fact, a program that has no terminating execution will always be partially correct. This is the case for example of the program while 1 do skip. The calculus offers no basis for a proof that a program terminates. It can be modified so as to account for total correctness of programs: It is achieved by amending the rule of iteration. We do not present it here and refer the interested reader to Apt []. Thus, if one first manages to prove that a program is deterministic, this trick works well enough to prove its total correctness. A general solution to the problem of total correctness exists in the realm of PDL. But we need to extend it a little. Pratt had already alluded in Pratt [b] that PDL is not expressive enough to capture the infinite looping of programs. A version of the conjecture in the variant of combinatory PDL was also proved in Gargov and Passy []. It is easy to see that in the Hoare calculus presented above, non termination can only come from the rule of iteration. Analogously, non termination of a PDL program can only come from the use of the unbounded iteration. Some variants Results concerning comparative power of expression, decidability, complexity, axiomatization and completeness of a number of variants of PDL obtained by extending or restricting its syntax and its semantics constitute the subject of a wealth of literature. We can only say so much and we will address just a few of these variants leaving out big chunks of otherwise important work in dynamic logic. It is interesting to observe that this assertion is untrue. Their argument actually can be generalized as follows. It is a construct that has been considered since the beginning of PDL. The converse construct allows us to express facts about states preceding the current one and to reason backward about programs. The addition of the converse construct does not change the properties of PDL in any significant way. By adding every instance of the following axiom schemas: See Parikh [] for details. Here, we summarize more results about RPDL and its connection with other variations on the notion of repeating programs. In Vardi and Stockmeyer [], an upper bound in non-deterministic exponential time was shown. If we add the converse operator of section 4. This is achieved by combining the techniques of Emerson and Jutla [] and Vardi [], as in Vardi []. This was shown in Harel and Sherman []. The proof involves the technique of filtration which is designed to collapse an LTS to a finite model while leaving invariant the truth or falsity of certain formulas. The set of equivalence classes of states thus obtained becomes the set of states of the filtrate model, and a transition is built appropriately over them. See Fischer and Ladner []. Indeed, the filtration must collapse some states of M and create some loops. There are other ways of making possible the assertion that a program can execute forever. For instance, Danecki [a] proposed a predicate sloop to qualify programs that can enter in strong loops, that is: SLPDL does not possess the finite model property. Another construct has been studied: Concerning the complexity theory of IPDL, difficulties appear when one considers the finite model property. We can thus adapt the formula of IPDL of section 4. In other words, IPDL does not possess the finite model property. In , Lange and Lutz [] gave a proof of a double exponential-time lower bound of the satisfiability problem for IPDL without tests by a reduction from the word problem of exponentially space-bounded alternating Turing machines. For this reason, the axiomatization of PDL with intersection was open until the complete proof system developed in Balbiani and Vakarelov []. Game Logic provides an additional program construct that dualizes programs, thus permitting to define the intersection of programs as the dual of the non-deterministic choice between programs. Conclusion This article has focused on propositional dynamic logic and some of its significant variants. The body of research on PDL is certainly instrumental in developing many logical theories of system dynamics. However, these theories are arguably out of the scope of the present article. Van Eijck and Stokhof [] is a more recent overview of topics making use of dynamic logic, addressing various themes that are of certain interest for philosophers: Recent books are going in much details on newer topics, such as dynamic logic of knowledge dynamic epistemic logic in Van Ditmarsch, Van Der Hoek and Kooi [], and the dynamic logic of continuous and hybrid systems differential dynamic logic in

DOWNLOAD PDF FIRST-ORDER DYNAMIC LOGIC

Platzer []. PDL was conceived primarily for reasoning about programs. There are many other applications of modal logic to reasoning about programs. Algorithmic logic is closer to PDL since it allows one to talk explicitly about programs. The reader is invited to consult the work studied in Mirkowska and Salwicki []. Temporal logics are now the chief logics in theoretical computer science and have a close connection with logics of programs. They allow one to express the temporal behavior of transition systems with a language that abstracts away from the labels hence the programs. See for instance Schneider [] for an overview of the foundations in this research area. North Holland Publishing Company, â€" Kooi, , Dynamic epistemic logic, Dordrecht: American Mathematical Society, 19â€" Center for the Study of Language and Information Publications. Salwicki, , Algorithmic Logic, Dordrecht: Proving Theorems for Complex Dynamics, Berlin:

Chapter 3 : Propositional Dynamic Logic (Stanford Encyclopedia of Philosophy)

First order dynamic logic (lecture notes in computer, enter your mobile number or email address below and we'll send you a link to download the free kindle app then you can start reading kindle books on your smartphone, tablet, or computer no.

Chapter 4 : CiteSeerX â€" Citation Query First-Order Dynamic Logic

Get Textbooks on Google Play. Rent and save from the world's largest eBookstore. Read, highlight, and take notes, across web, tablet, and phone.

Chapter 5 : Dynamic logic (modal logic) - Wikipedia

First-Order Dynamic Logic by David Harel starting at $ First-Order Dynamic Logic has 2 available editions to buy at Half Price Books Marketplace.

Chapter 6 : The expressibility of first order dynamic logic - [PDF Document]

First-Order Dynamic Logic. Editors; David Harel; Book. 68 Citations; Modale Logik Programmiersprache Programmierung computation logic programming language semantics.

Chapter 7 : KeYmaera: A Hybrid Theorem Prover for Hybrid Systems

As one of the premier rare book sites on the Internet, Alibris has thousands of rare books, first editions, and signed books available. With one of the largest book inventories in the world, find the book you are looking for. To help, we provided some of our favorites. With an active marketplace of.

Chapter 8 : Prof. David Harel - Books

Dynamic logic is an extension of modal logic originally intended for reasoning about computer programs and later applied to more general complex behaviors arising in linguistics, philosophy, AI, and other fields.