UNIVERSITÀ DEGLI STUDI DI BARI ALDO MORO

DIPARTIMENTO DI INFORMATICA

Dottorato di ricerca in Informatica e Matematica
XXIX ciclo

Progetto di ricerca

Dottorando: Dott. Gennaro Vessio
Tutor: Dott. Alessandro Bianchi

Coordinatore
Prof. Donato Malerba

Firma del dottorando _____

Firma del tutor _____

1) Titolo della ricerca:

Una metodologia per l'analisi di sistemi mediante Abstract State Machine

2) Area nella quale si inquadra la ricerca:

Metodi formali

3) Obiettivi della ricerca

Al fine di realizzare sistemi informatici critici e complessi, spesso distribuiti, e i cui eventuali malfunzionamenti potrebbero causare conseguenze disastrose, un'ampia comunità, sia accademica (si veda, ad esempio, [23], [19] e [33]) che industriale (si veda, ad esempio, [26]), propone il ricorso a metodi formali. Secondo tale approccio, il sistema da realizzare viene dapprima modellato formalmente, e solo successivamente implementato. Questo approccio permette di svolgere ad alti livelli di astrazione e nelle fasi iniziali dello sviluppo l'analisi di proprietà computazionali di interesse per il sistema: *starvation-freedom*, *deadlock-freedom*, *reachability*, *reversibility*, ecc. [22]. L'indubbio vantaggio di tale analisi, anticipata rispetto alla realizzazione, riguarda sia la sua correttezza, provata rigorosamente grazie al formalismo, sia i costi, più contenuti rispetto all'analisi, svolta *ex-post*, sul sistema finale.
Diversi formalismi, come ad esempio le reti di Petri [11], sono corredati da definizioni formali di proprietà, intrinseche al formalismo stesso, tali per cui le attività di verifica formale possono essere condotte in maniera sistematica. Tuttavia per altri formalismi, in particolare le Abstract State Machine (ASM) [17], il problema di definire una metodologia intrinseca al formalismo per lo studio delle proprietà rappresenta ancora un problema aperto. Il presente progetto di ricerca intende affrontare tale problematica definendo precisamente, in termini di ASM, proprietà computazionali d'interesse, così da poter poi sviluppare una metodologia, integrata al formalismo stesso, per la loro analisi. Più nello specifico, la ricerca intende concentrarsi sulle classi di proprietà note come proprietà di *liveness* e di *safety*; informalmente, le prime implicano che "qualcosa di desiderato prima o poi deve accadere", le seconde che "qualcosa di indesiderato non deve mai accadere" [22]. L'analisi di tali classi permette infatti di caratterizzare l'eventuale presenza di situazioni di *deadlock* e di *starvation*; di garantire la raggiungibilità di determinati stati e di prevedere la reversibilità della computazione a situazioni desiderate.

4) Motivazioni della ricerca

La possibilità di disporre di metodologie per l'analisi delle proprietà d'interesse per i sistemi in via di sviluppo è un'esigenza da sempre avvertita dagli informatici. Questa esigenza, data la sempre maggiore connessione tra Information Technology e Communication Technology, che ha portato negli ultimi anni gli informatici ad affrontare il tradizionale paradigma della computazione in stretta relazione con quello della comunicazione, è ancora più avvertita nel contesto di sistemi distribuiti critici e complessi. Ne sono esempi le Mobile Ad-hoc NETwork (MANET) [1], i sistemi Grid [6] e i sistemi Cloud [27] (per una trattazione più approfondita si rimanda a [31]).
Nel contesto delle ASM, trattandosi di un formalismo Turing-equivalente [17], una soluzione algoritmica all'analisi di proprietà è irrealizzabile, dal momento che la sua ipotetica esistenza

violerebbe l'indecidibilità dell'*halting problem*. Ciononostante, in letteratura, sono state proposte diverse soluzioni ad hoc, per casi specifici, basate sul ricorso a tecniche di *model checking* [4]. Si tratta tuttavia di approcci ibridi che soffrono di alcune limitazioni: la perdita di potere espressivo, derivante dalla traduzione della ASM in esame in un automa a stati finiti (o sue varianti); e il ricorso a formalismi alternativi, tipicamente logiche temporali, per esprimere le proprietà d'interesse. La metodologia che la ricerca intende sviluppare vuole invece essere focalizzata interamente all'interno del formalismo delle ASM. Per via, infatti, della poca familiarità degli sviluppatori di software con l'uso di approcci ibridi, l'esigenza di disporre di specifiche "operazionali" delle proprietà, piuttosto che "dichiarative", in quanto basate su logiche temporali, è un problema riconosciuto (si veda, ad esempio, [3], [21]).
Si è scelto di concentrare l'attenzione della ricerca sul formalismo delle ASM per svariati motivi: l'espressività del formalismo, capace di catturare il comportamento sequenziale [17] e concorrente [8] di sistemi di qualsivoglia complessità a qualunque livello di astrazione; dal punto di vista metodologico, l'importanza delle ASM in un metodo di sviluppo del software, il Metodo ASM [10], utilizzato con successo nell'analisi di sistemi con elevati gradi di criticità e complessità in svariati domini applicativi; l'esistenza di tool, come AsmL [18], CoreASM [13] e ASMETA [16], che permettono di tradurre specifiche ASM in codice eseguibile e quindi di condurre simulazioni; infine, la centralità delle ASM in un recente dibattito sul tema "Cos'è un algoritmo?" [32]. In risposta al quesito sollevato, il dibattito ha condotto a due proposte distinte: quella di Moschovakis, che propone la definizione di "ricursore" come definizione formale di algoritmo, e quella di Gurevich, che, al contrario, propone la definizione di ASM.

5) Stato dell'arte

La definizione di una metodologia per l'analisi di proprietà integrata interamente all'interno di un unico formalismo è un problema noto. In letteratura è stato affrontato nell'ambito di altri formalismi; ne sono esempi le reti di Petri [20] e il CSP [9]. Le reti di Petri [11] si sono rivelate particolarmente adatte alla rappresentazione dei comportamenti dinamici di sistemi discreti che tipicamente esibiscono attività asincrone e concorrenti. Il CSP di Hoare [19] appartiene invece alla famiglia delle *process algebra*; si tratta, dunque, di un formalismo che enfatizza gli aspetti di interazione, in altre parole gli eventi osservabili del sistema, piuttosto che il comportamento interno di ciascuna delle sue componenti. A differenza delle reti Petri, CSP è stato impiegato principalmente per l'analisi di sistemi "sicuri" (si veda ad esempio [28]).
Informalmente, le ASM sono automi a stati finiti con stati astratti [10]. Per stato astratto s'intende una struttura algebrica arbitrariamente complessa, composta di funzioni ciascuna con un proprio dominio e codominio. La transizione da uno stato all'altro, invece, è governata da regole nella forma *if condition then updates*, tali che se la *condition*, espressa in logica del prim'ordine, è soddisfatta, allora una serie di aggiornamenti del valore corrente delle funzioni è eseguita sulla base di *updates*. Il framework ASM, ad oggi, permette attività di verifica formale sia manuale che automatica. In [10], per esempio, sono riportati numerosi esempi di verifica manuale. Dal momento che le ASM sono macchine eseguibili che si prestano all'adozione di metodi di induzione, tali prove sono spesso formulate in maniera matematica. Tipicamente, gli autori individuano le condizioni che sussistono nello stato iniziale della macchina e, attraverso l'esplorazione di tutti i possibili cammini di esecuzione della ASM in esame, verificano se le condizioni desiderate continuano a valere e/o le si riscontra negli stati finali della computazione. Un esempio di questa strategia di verifica è la prova di correttezza del protocollo di autenticazione Kerberos [5]. Stärk e Nanchen [30], invece, propongono una formalizzazione delle ASM nei termini della logica del prim'ordine. Ciò ha permesso agli autori di esprimere formalmente due proprietà intrinseche al formalismo: vale a dire le cosiddette "coerenza" e "gerarchia". In [15], infine, viene proposto un sistema formale di prova

basato sulla logica di Hoare. Tuttavia, in ognuno di questi casi, le proprietà d'interesse sono sempre espresse in maniera fortemente dipendente dal dominio applicativo d'interesse.
Di contro, il problema della verifica automatica delle ASM è stato posto per la prima volta da Marc Spielmann nella sua tesi di dottorato [29]. L'autore ha dimostrato che il problema è decidibile solo se si considerano classi ristrette di ASM, quelle che l'autore chiama *sequantial nullary* ASM, il cui potere espressivo è però notevolmente ridotto. In [12], [14] e [2], in cui vengono proposti model checker per ASM integrati ai già citati tool AsmL, CoreASM ed ASMETA, rispettivamente, gli autori applicano un approccio basato su model checking. Il model checking [4] è una tecnica volta alla verifica automatica di modelli formali: il modello è espresso sotto forma di automi a stati finiti o loro varianti, mentre le proprietà d'interesse sono specificate in logica temporale. L'approccio si basa sulla traduzione della ASM in esame nell'input richiesto dal model checker; tuttavia tale passaggio equivale ad una riduzione della ASM ad un automa a stati finiti, e quindi comporta una inevitabile perdita di potere espressivo.
L'esigenza di definire, nell'ambito delle ASM, una metodologia per lo studio delle proprietà slegata dall'adozione di approcci ibridi è stata sollevata in [3]. Gli autori adducono come motivazione la maggiore familiarità degli sviluppatori di software con l'uso di specifiche "operazionali" delle proprietà, anziché con l'uso di specifiche "dichiarative" quali appunto le logiche temporali. Anche in un altro recente lavoro [21], gli autori enfatizzano la necessità di definire specifiche "operazionali" delle proprietà e, inoltre, di slegarsi, laddove possibile, dalla conoscenza di dominio, in genere molto specifica. Tuttavia gli autori non trattano le ASM, bensì un formalismo di recente introduzione basato sulla teoria dei grafi.

6) Approccio al problema

S'intende affrontare il problema oggetto di ricerca ricorrendo a due tipi di approccio fortemente interconnessi l'uno all'altro: analisi concettuale e studio della sua applicazione.
L'approccio concettuale è volto a indagare il fondamento teorico delle principali proprietà che già si ritrova nel contesto di altri formalismi; e di ridefinirlo all'interno del formalismo delle ASM, adattandolo alle caratteristiche specifiche di quest'ultimo. Il soddisfacimento di determinate proprietà dipende strettamente dal comportamento esibito dai modelli per i quali quelle proprietà valgono; di conseguenza, l'approccio mira a studiare e isolare quali sono le caratteristiche ASM che garantiscono il soddisfacimento di determinate proprietà, e quali invece le caratteristiche ASM che ne impediscono il soddisfacimento.
Dall'altro lato, l'approccio applicativo mira ad applicare i risultati concettuali a modelli ASM di sistemi critici e complessi reali. Ogni approccio si nutre dei risultati ottenuti dall'altro: l'analisi di modelli di sistemi reali suggerisce caratteristiche che dovranno essere generalizzate nell'ambito concettuale al fine di formulare precise definizioni; le definizioni formali, d'altro canto, permettono analisi più accurate dei modelli dei sistemi presi in considerazione.
La parte finale della ricerca vorrà sintetizzare le esperienze svolte in una metodologia per l'analisi delle proprietà di modelli ASM applicabile alla più ampia classe possibile di sistemi.

7) Ricadute applicative

Dal punto di vista della ricerca di base, una soluzione al problema di definire una metodologia per l'analisi di proprietà nell'ambito delle ASM ha un duplice impatto. Da un lato, il miglioramento, da un punto di vista teorico, della precisione del framework ASM, in quanto arricchito da definizioni di proprietà espresse nei termini del formalismo stesso. Dall'altro lato, il miglioramento, da un punto di vista applicativo, dell'applicabilità del Metodo ASM, in quanto arricchito da una

metodologia per lo studio delle suddette proprietà utilizzabile in maniera sistematica. Inoltre, l'oggetto della ricerca suscita interesse poiché può fornire ulteriore evidenza empirica ad una tesi proposta di recente da alcuni ricercatori [7], [24]. Secondo tale tesi, l'implementazione di un sistema ottenuta mediante "raffinamento contestuale" [24], ossia mediante la realizzazione pedissequa del sistema sulla base di un suo modello più astratto, preserva esattamente le stesse proprietà e comportamenti già soddisfatti dal modello.
Da un punto di vista, invece, industriale, una tale soluzione è d'interesse per almeno due motivazioni: il supporto ad analisi *ex-ante*, dal momento che l'identificazione di eventuali comportamenti anomali o malfunzionamenti, a priori dell'implementazione, riduce notevolmente i costi di manutenzione; e, parallelamente, il supporto ad analisi *ex-post*, in quanto l'applicazione dell'approccio formale a sistemi pre-esistenti spesso è in grado di condurre alla realizzazione di varianti volte al miglioramento di alcuni aspetti computazionali rivelatisi insoddisfacenti e/o vulnerabili. Ne è un esempio la variante più robusta del protocollo di Needham-Schroeder proposta da Lowe grazie ad analisi condotte mediante CSP [25].
Fra i possibili scenari applicativi della metodologia oggetto di ricerca, come già accennato nella Sezione 4, ritroviamo MANET [1], sistemi Grid [6] e sistemi Cloud [27]. Si tratta di sistemi distribuiti critici e complessi tipicamente caratterizzati da alti gradi di modularità e interconnessione e le cui componenti cooperano (in taluni casi concorrono) per il raggiungimento di obiettivi computazionali comuni. Per sistemi di questo tipo, l'analisi di quegli aspetti e quelle caratteristiche specifiche che potrebbero causare l'insorgere di problemi, quali *deadlock*, *starvation*, stati computazionali irraggiungibili o irreversibili, ecc., benché ardua, riveste evidentemente un ruolo chiave.

8) Fasi del progetto

Il progetto di ricerca si articola in tre fasi, ciascuna delle quali svolta nell'arco del triennio di dottorato.

1. Studio dello stato dell'arte. L'obiettivo è padroneggiare lo stato dell'arte della letteratura di riferimento. In particolare: il fondamento teorico della verifica formale di sistemi software; le diverse classificazioni di proprietà già proposte dai ricercatori; l'approccio allo sviluppo di metodologie per la verifica delle proprietà già applicato nell'ambito di altri formalismi più consolidati (come le già citate reti di Petri [11]); i lavori di ricerca che hanno affrontato il problema nel caso specifico delle ASM.
2. Definizione di proprietà. L'obiettivo è definire formalmente proprietà computazionali d'interesse in termini di ASM. Questa fase necessita lo studio di modelli ASM di sistemi che esibiscono le proprietà desiderate, al fine di isolare quelle caratteristiche ASM tali da garantirne il soddisfacimento. Per questo motivo questa fase si avvale della successiva fase (3).
3. Studio delle proprietà in modelli di sistemi reali. L'obiettivo è sviluppare una metodologia integrata al framework ASM per lo studio e l'analisi delle proprietà precedentemente definite. Dal momento che questa fase necessita di generalizzare le caratteristiche ASM precedentemente identificate e isolate, essa si avvale delle definizioni di proprietà, eventualmente preliminari, di cui al punto (2).
4. Applicazione della metodologia proposta. S'intende applicare la metodologia di cui al punto (3) ad altri modelli di sistemi reali con lo scopo di valutare i risultati ottenuti dalla ricerca.

Le quattro fasi abbracceranno trasversalmente l'intero triennio di dottorato. Tuttavia è possibile identificare, benché in maniera approssimativa, momenti di maggiore concentrazione. In particolare: la fase 1 sarà maggiormente concentrata nel primo anno di dottorato; le fasi 2 e 3 nel secondo anno; la fase 4 nella prima metà del terzo anno. La seconda metà del terzo anno sarà, invece, dedicata alla stesura della tesi finale.

9) Valutazione dei risultati

Al fine di valutare i risultati della ricerca, s'intende applicare la metodologia proposta a casi di studio reali critici e complessi. Come accennato nella Sezione 6, alcuni casi di studio possono essere proficuamente impiegati per l'individuazione di quei pattern che isolano le caratteristiche ASM tali per cui determinate proprietà sono soddisfatte. Altri casi di studio, invece, rappresenteranno il banco di prova per sperimentare la metodologia sviluppata, e quindi valutarne l'effettiva applicabilità. Ci si aspetta che la metodologia proposta:

- sia in grado di stabilire se un determinato modello ASM sotto analisi soddisfa o meno le proprietà desiderate;
- sia in grado di identificare, in caso negativo, le aree del modello soggette al rischio di violazione delle proprietà non soddisfatte.

10) Riferimenti bibliografici

1. Agrawal, D.P., Zeng, Q.A.: Introduction to Wireless and Mobile Systems. Thomson Brooks/Cole (2003)
2. Arcaini, P., Gargantini, A., Riccobene, E.: AsmetaSMV: AWay to Link High-Level ASM Models to Low-Level NuSMV Specifications. In: 2th International Conference on Abstract State Machines, Alloy, B and Z, pp. 61–74, Springer-Verlag (2010)
3. Arcaini, P., Gargantini, A., Riccobene, E.: CoMA: Conformance Monitoring of Java Programs by Abstract State Machines. In: 2nd International Conference on Runtime Verification, pp. 223–238, Springer-Verlag (2012)
4. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press (2008)
5. Bella, G., Riccobene, E.: Formal Analysis of the Kerberos Authentication System. Journal of Universal Computer Science, 3(12), pp. 1337–1381 (1997)
6. Berman, F., Fox, G., Hey, A.J.G.: Grid Computing: Making the Global Infrastructure a Reality. John Wiley & Sons (2003)
7. Bianchi, A., Pizzutilo, S.: A Coloured Nested Petri Nets Model for Discussing MANET Properties. International Journal of Multimedia Technology, 3(2), pp. 38–44 (2013)
8. Blass, A., Gurevich, Y.: Abstract State Machines Capture Parallel Algorithms. ACM Transaction on Computational Logic, 4(4), pp. 578–651 (2003)
9. Bordeaux, L.: CSP Properties for Quantified Constraints: Definitions and Complexity. In: AAAI-2005, pp. 360–365 (2005)
10. Börger, E., Stärk, R.: Abstract State Machines: A Method for High-Level System Design and Analysis. Springer-Verlag (2003)
11. David, R., Alla, H.: Discrete, Continuous and Hybrid Petri Nets. Springer-Verlag (2005)
12. Del Castillo, G., Winter, K.: Model Checking Support for the ASM High-Level Language. In: Graf, S., Schwartzbach, M., (eds.), 6th International Conference TACAS 2000, 1785, pp. 331–346, LNCS, Springer-Verlag (2000)
13. Farahbod, R., Gervasi, V., Glässer, U.: CoreASM: An Extensible ASM Execution Engine. Fundamenta Informaticae, 77(1-2), pp. 71–103 (2007)
14. Farahbod, R., Glässer, U., Ma, G.: Model Checking CoreASM Specifications. In: Prinz, A. (ed.), ASM '07, the 14th International ASM Workshop (2007)
15. Gabrisch, W.: A Hoare-Style Verification Calculus for Control State ASMs. In: 5th Balkan Conference on Informatics, pp. 205–210 (2012)
16. Gargantini, A., Riccobene, E., Scandurra, P.: Model-Driven Language Engineering: The ASMETA Case Study. In: 3rd International Conference on Software Engineering Advances, pp. 373–378 (2008)
17. Gurevich, Y.: Sequential Abstract State Machines Capture Sequential Algorithms. ACM Transactions on Computational Logic, 1(1), pp. 77–111 (2000)
18. Gurevich, Y., Rossman, B., Schulte, W.: Semantic Essence of AsmL. Theoretical Computer Science, 342(3), pp. 370–412 (2005)
19. Hoare, C.A.R.: Communicating Sequential Processes. Prentice Hall (2004)

20. Jensen, K., Kristensen, L.M., Wells, L.: Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. International Journal on Software Tools for Technology Transfer, 9(3-4), pp. 213–254 (2007)
21. Kai, K., Desel, J.: Checking Soundness of Business Processes Compositionally Using Symbolic Observation Graphs. In: 32th International Conference on Formal Techniques for Distributed Systems, pp. 67–83, Springer-Verlag (2012)
22. Kindler, E.: Safety and Liveness Properties: A Survey. EATCS Bulletin, No. 53, pp. 268–272 (1994)
23. Lamport, L.: While waiting for the millennium: Formal specification and verification of concurrent systems now. In: International Conference on Concurrency, pp. 1–3 (1988)
24. Liang, H., Hoffmann, J., Feng, X., Shao, Z.: Characterizing Progress Properties of Concurrent Objects via Contextual Refinements. In: 24th International Conference on Concurrency Theory, pp. 227–241 (2013)
25. Lowe, G.: Hierarchy of Authentication Specifications. In: Computer Security Foundations Workshop, pp. 31–43 (1997)
26. Microsoft Research. `http://research.microsoft.com/en-us/`
27. Rimal, B.P.: A Taxonomy and Survey of Cloud Computing Systems. In: 5th International Joint Conference on INC, IMS and IDC, pp. 44–51 (2009)
28. Schneider, S.: Security Properties and CSP. In: IEEE Symposium on Security and Privacy, pp. 174–187 (1996)
29. Spielmann, M: Abstract State Machines: Verification Problems and Complexity. PhD thesis, RWTH Aachen (2000)
30. Stärk, R., Nanchen, S.: A Logic for Abstract State Machines. Computer Science Logic, pp. 217–231, Springer (2001)
31. Tanenbaum, A., Van Steen, M.: Distributed Systems: Principles and Paradigms, 2nd edition. Pearson Education (2007)
32. Vardi, M.: What is an Algorithm? Communications of the ACM, 55(3), p. 5 (2012)
33. Winskel, G: The Formal Semantics of Programming Languages: An Introduction. The MIT Press (1993)

11) Eventuali referenti esterni al Dipartimento