Summary: This paper presents a brief account of some of the my early research interests. This historical account starts from my laurea thesis on Signal Theory and my master thesis on Computation Theory. It recalls some results in Combinatory Logic and Term Rewriting Systems. Some other results concern Program Transformation, Parallel Computation, Theory of Concurrency, and Proof of Program Properties. My early research activity has been mainly done in cooperation with Andrzej Skowron, Anna Labella, and Maurizio Proietti.

For the entire collection see [Zbl 1466.68013].

MSC:

68N30 Mathematical aspects of software engineering (specification, verification, metrics, requirements, etc.)
68Q60 Specification and verification (program logics, model checking, etc.)

Full Text: arXiv Link

References:

[1] flipcheck(X, Y ) ← flip(X, Y ), check(Y )
[2] flip((L, N, R), t(FR, N, FL)) ← flip(L, FL), flip(R, FR)
[3] check(t(N )) ← nat(N )
[4] check(t(L, N, R)) ← nat(N ), check(L), check(R) uses the continuation passing style: 8 flipcheck(X, Y ) ← newp(X, Y, G, true, G) 9 newp(t(N ), l(N ), G) 10 newp(t(L, N, R), t(FR, N, FL), G) 11 newp(L, FL, U, C, D) 12 newp(R, FL, U, D) 13 newp(G, nat1(C, C), D)) (s(N ), C) ← nat1(N, C) For the predicate eq1 we assume that: 14 eq1(X, Y, C) 15 (X = Y ) 16 C)
[5] production(x, [a, b, x]) ← 11. production(x, [a, b, x]) ← 12. word([ ]) ← 13. word([A|W ]) ← terminal(A), word(W ) 14. append([ ],Ys,Ys) ← 15. append([A|Xs],Ys,[A|Zs]) ← append(Xs,Ys, Zs)
[6] person is dead. (s4) If the gun is loaded, then it is abnormal that after a shoot event the person is alive. (s5) Inertia Axiom: If a fact F holds in a situation S and it is not abnormal that F holds after the event E following S, then F holds also after the event E. The following locally stratified program YSP formalizes the above statements. A similar
[7] ab(alive, shoot, S) ← holds(loaded, S), event(E), holds(F, S), ¬ ab(F, E, S) 11. holds(F, [E|S]) ← fact(F ), event(E), holds(F, S), ¬ ab(F, E, S) 12. append([ ],Ys,Ys) ← 13. append([A|Xs],Ys,[A|Zs]) ← append(Xs,Ys, Zs)
[8] By applying SLDNF-resolution [38], Apt and Bezem showed that holds(dead, [shoot, wait, load]) is true in the perfect model of program YSP. Now we consider a property Γ which cannot be shown by SLDNF-resolution (see [58]):
[9] Γ (holds(dead, S) → holds(dead, S))
[10] Property Γ means that the fact that the person is dead in the current situation S implies that in the past there was a load event followed, possibly not immediately, by a shoot event. Thus, since time progresses ‘to the left’, S is a list of events of the form: [ . . . , shoot, . . . , load, . . . ]
[11] In the first step of our two step verification method [38, Section 2.3], we apply the Lloyd-Topor transformation [38, page 113] starting from the statement: g ← Γ (where g is a new predicate name) and we derive the following clauses: 14. g ← ¬new1 15. new1 ← holds(dead, S), ¬new2(S) 16. new2(S) ← append(S, S) 17. append(S, S) 18. [load[S0], S])
[12] At the end of the second step, after a few iterations of the unfold-definition-folding strategy and after the deletion of all definitions of predicates which are not required by g, we are left with the single clause: g ←. Details can be found in [58].
[13] Since g holds in the (perfect model of the) final program, we have that property Γ holds in the (perfect model of the) final program. Thus, Γ holds also in the initial program made out of clauses 1-13. Much more recently we have explored some verification techniques based on the transfer-mation of constrained Horn clauses, also in the case of imperative and functional programs [19] and in the case of business processes (see, for instance, [17]). This recent work has been done in cooperation with Emanuele De Angelis and Fabio Fioravanti. They also have been working and still work in the implementation and development of an automatic transformation and verification tool [18], which was originally set up by Ornella Aioni and Maurizio Proietti.

Edited by FIZ Karlsruhe, the European Mathematical Society and the Heidelberg Academy of Sciences and Humanities © 2022 FIZ Karlsruhe GmbH