

CLAIMS

1. A method for securing a first program with a computer system including a processor and memory, the first program comprising a finite number of program points and evolution rules associated with the program points and defining the passage from one program point to another program point, ~~comprises~~the method comprising:

- ~~a definition of~~defining, with the computer system, a plurality of exit cases in a non-transitory computer readable medium and, when a second program is used in the definition of the first program, the second program defining an executable subroutine of the first program and for each exit case of the second program comprising a finite number of program points and evolution rules associated with the program points and defining, ~~definition of~~ a branching toward a specific program point of the first program or a declaration of branching impossibility; and
- ~~a definition of~~defining, with a computer system, ~~of~~ a set of local properties to be proven, each associated with one or more ~~of the constitutive elements~~evolution rules of the first program, said set of local properties comprising the branching impossibility as a particular local property; and
- ~~establishment~~establishing, with a computer system, a ~~of the~~ formal proof of the set of local properties.

2. The method according to claim 1, ~~characterized in that~~wherein the evolution rules, the exit ~~points~~cases and the branchings defining a tree structure of execution traces or of logic traces, a prover uses the created tree structure to propagate constraints in points of the first program and uses ~~the~~at least one transitive closure of explicit or implicit congruences in order to prove at least one of the properties of the first program.

3. The method according to claim 2, ~~characterized in that~~wherein the first program comprising at least a loop, ~~a~~the prover automatically generates an invariant at a program point by using potentially approximate versions of the execution ~~routes~~traces leading thereto, and wherein the execution traces ~~parts~~ representing the execution ~~of~~of the at least a loops have been replaced by approximate steps.

4. The method according to claim 2, ~~characterized in that~~wherein at least a branching declared as "impossible" is proven by analyzing the execution routes or the tree structure ~~routes of the execution traces~~ leading to the transition.
5. The method according to claim 2, ~~wherein characterized in that~~ the securing of a first program is based on the securing of a third program having different functionalities from the first program, and/or not merely being an abstraction of the first program .
6. The method according to claim 5, ~~further comprising establishing characterized in that~~ oriented correspondences, ~~called or~~ sufficiency relationships, between ~~at least one of the a first~~ local ~~properties property~~ of the first program and ~~at least one of the a~~ set of local properties of the third program ~~are established~~, such that ~~for~~ proving a the first local property of the first program, ~~it~~ is sufficient to prove the set of properties of the third program ~~which are in correspondence with this property of the first program~~, whenever such correspondences exist.
7. The method according to claim 6, ~~characterized wherein in that~~ the third program is modified by a transformation making it possible to obtain the first program, such that the securing of the first program is based on the securing of the third program .
8. The method according to claim 6, ~~wherein characterized in that~~ the first program is modified by a transformation making it possible to obtain the third program, such that the securing of the first program is based on the securing of the third program-.
9. The method according to claim 6, ~~characterized in that~~wherein the first program is modified by a transformation making it possible to obtain the third program and draw up a plurality of sufficiency relationships oriented between at least one of the respective properties of the first and third programs.
10. The method according to claim 6, ~~characterized in that~~wherein the third program is modified by a transformation making it possible to obtain the first program and to draw up a plurality of sufficiency relationships oriented between at least one of the respective properties of the first and third programs.

11. The method according to claim 5, ~~characterized in that~~wherein at least one of the variables of the third program is instantiated using values taken by the variables of the first program to draw up at least a sufficiency relationship between the first program and the third program, or as a way of securing the first program.

12. The method according to ~~any one of the preceding claims~~claim 1, ~~characterized in that~~wherein at least one set of local ~~property~~properties is defined by a fourth program.

13. A computer software product comprising software instructions to implement the method according to any one of claims 1 to 12 when said computer software product is executed on a computer.

14. The computer software product according to claim 13, characterized in that it comprises a user interface adapted for enabling a user to define at least one of the plurality of exit cases and associated branchings, and ~~or~~ set of local properties to be proven.

15. ~~he~~The computer software product according to claim 14, characterized in that the user interface is adapted to define an impossible branching.

16. The computer software product according to claim 14 or 15, characterized in that the user interface is adapted to define entry conditions of a particular program to use ~~or~~ call at a given point and thus limit the at least one exit case of the called program .

17. The computer software product according to claim 13 that generate implementation code in a way that takes advantage of the ~~fact that some of the impossible transitions or local properties have been proven so as~~formal branch proof of branching impossibility to optimize the generated code-.

18. A method for securing a first program, the first program comprising a finite number of program points and evolution rules associated with the program points and defining the passage from one program point to at least another program point, ~~comprises~~the method comprising:

- ~~a definition of defining~~ a plurality of exit cases and, when a second program is used in the definition of the first program, the second program defining an executable subroutine of the first program and ~~for each exit case of~~ the second program, comprising a finite number of program points and evolution rules associated with the program points and defining the passage from one program point to another program point until at least one exit case, ~~definition of defining~~ a branching toward a specific program point of the first program or a declaration of branching impossibility; and

- ~~a definition of defining~~ a set of local properties to be proven, each associated with one or more ~~of the constitutive elements~~ program points and evolution rules of the first program, said set of properties comprising the branching impossibility as a particular property;

- establishment of the formal proof of the set of local properties-

wherein the first program comprising at least a loop, a prover automatically generates an invariant at a program point by using potentially approximate versions of the execution routes leading thereto, wherein the ~~parts representing the execution routes of the at least a execution of~~ loops have been replaced by approximate steps.

19. The method according to claim 18, ~~characterized in that~~ wherein at least a branching declared as "impossible" is proven by analyzing the execution routes or the tree structure of the execution traces or of logic traces ~~routes~~ leading to the transition.

20. A method for securing a first program in a non-transitory computer readable medium with a computer system including a processor and memory, the first program comprising a finite number of program points and evolution rules associated with the program points and defining the passage from one program point to at least another program point, comprises:

- ~~a definition of defining, with the computer system, of~~ a plurality of exit cases in the non-transitory computer medium and, when a second program is used in the definition of the first program, the second program defining an executable subroutine of the first program and the second program comprising a finite number of

~~program points and evolution rules associated with the program points and defining the passage from one program point to another program point until at least one exit case for each exit case of the second program, definition defining of~~ a branching toward a specific program point of the first program or a declaration of branching impossibility; and

- ~~a definition of defining, with the computer system,~~ a set of local properties to be proven, each associated with one or more ~~of the constitutive elements~~program points and evolution rules of the first program, said set of local properties comprising the branching impossibility as a particular local property;
- ~~establishment establishing, with the computer system, of the~~ formal proof of the set of properties,

wherein the securing of a first program is based on the securing of a third program having different functionalities from the first program, and/or not merely being an abstraction of the first program.

21. The method according to claim 20, ~~characterized in that oriented correspondences~~and further comprising oriented correspondences, ~~]~~ called or sufficiency relationships, between ~~at least one of the~~a first local ~~properties property~~ of the first program and ~~at least one of the~~a set of local properties of the third program ~~are established~~, such that ~~for proving a the first~~ local property of the first program, ~~it is sufficient to prove the set of local properties of the third program which are in correspondence with this property of the first program,~~ whenever such correspondences exist.

22. The method according to claim 21, ~~characterized in that~~ wherein the third program is modified by a transformation making it possible to obtain the first program, such that the securing of the first program is based on the securing of the third program-.

23. The method according to claim 21, ~~characterized in that~~wherein the first program is modified by a transformation making it possible to obtain the third program, such that the securing of the first program is based on the securing of the third program-.

24. The method according to claim 21, ~~characterized in~~wherein ~~that~~ the first program is modified by a transformation making it possible to obtain the third program and draw up a plurality of sufficiency relationships oriented between at least one of the respective properties of the first and third programs.

25. The method according to claim 21, ~~characterized in that~~wherein the third program is modified by a transformation making it possible to obtain the first program and to draw up a plurality of sufficiency relationships oriented between at least one of the respective properties of the first and third programs-.

CLAIMS

1. A method for securing a first program, the first program
5 comprising a finite number of program points and
evolution rules associated with the program points and
defining the passage from one program point to another
program point, comprises:

- a definition of a plurality of exit cases and,
10 when a second program is used in the definition of the
first program, for each exit case of the second program,
definition of a branching toward a specific program point
of the first program or a declaration of branching
impossibility; and

- 15 • a definition of a set of properties to be
proven, each associated with one or more of the
constitutive elements of the first program, said set of
properties comprising the branching impossibility as a
particular property;

- 20 • establishment of the formal proof of the set of
properties.

2. The method according to claim 1, characterized in that the
evolution rules, the exit points and the branchings defining a
25 tree structure of execution traces or of logic traces, a
prover uses the created tree structure to propagate
constraints in points of the first program and uses the
transitive closure of explicit or implicit congruences in
order to prove at least one of the properties of the first
30 program.

3. The method according to claim 2, characterized in that the
first program comprising at least a loop, a prover
automatically generates an invariant at a program point by

using potentially approximate versions of the execution routes leading thereto, wherein the parts representing the execution of loops have been replaced by approximate steps.

5 4. The method according to claim 2, characterized in that at least a branching declared as "impossible" is proven by analyzing the tree structure routes leading to the transition.

10 5. The method according to claim 2, characterized in that the securing of a first program is based on the securing of a third program having different functionalities from the first program, and/or not merely being an abstraction of the first program.

15 6. The method according to claim 5, characterized in that oriented correspondences, called sufficiency relationships, between at least one of the local properties of the first program and at least one of the local properties of the third program are established, such that for proving a local
20 property of the first program, it is sufficient to prove the set of properties of the third program which are in correspondence with this property of the first program, whenever such correspondences exist.

25 7. The method according to claim 6, characterized in that the third program is modified by a transformation making it possible to obtain the first program, such that the securing of the first program is based on the securing of the third program.

30 8. The method according to claim 6, characterized in that the first program is modified by a transformation making it possible to obtain the third program, such that the securing

of the first program is based on the securing of the third program.

9. The method according to claim 6, characterized in that the first program is modified by a transformation making it possible to obtain the third program and draw up a plurality of sufficiency relationships oriented between at least one of the respective properties of the first and third programs.

10. The method according to claim 6, characterized in that the third program is modified by a transformation making it possible to obtain the first program and to draw up a plurality of sufficiency relationships oriented between at least one of the respective properties of the first and third programs.

11. The method according to claim 5, characterized in that at least one of the variables of the third program is instantiated using values taken by the variables of the first program to draw up at least a sufficiency relationship between the first program and the third program, or as a way of securing the first program.

12. The method according to any one of the preceding claims, characterized in that at least one local property is defined by a fourth program.

13. A computer software product comprising software instructions to implement the method according to any one of claims 1 to 12 when said computer software product is executed on a computer.

14. The computer software product according to claim 13, characterized in that it comprises a user interface adapted

for enabling a user to define exit cases and associated branchings, and/or local properties to be proven.

15. The computer software product according to claim 14,
5 characterized in that the user interface is adapted to define an impossible branching.

16. The computer software product according to claim 14 or 15,
10 characterized in that the user interface is adapted to define entry conditions of a particular program to use/call at a given point and thus limit the exit case of the called program.

17. The computer software product according to claim 13 that
15 generate implementation code in a way that takes advantage of the fact that some of the impossible transitions or local properties have been proven so as to optimize the generated code.

20 18. A method for securing a first program, the first program comprising a finite number of program points and evolution rules associated with the program points and defining the passage from one program point to at least another program point, comprises:

- 25 • a definition of a plurality of exit cases and, when a second program is used in the definition of the first program, for each exit case of the second program, definition of a branching toward a specific program point of the first program or a declaration of branching impossibility; and
- 30 • a definition of a set of properties to be proven, each associated with one or more of the constitutive elements of the first program, said set of

properties comprising the branching impossibility as a particular property;

- establishment of the formal proof of the set of properties,

5 wherein the first program comprising at least a loop, a prover automatically generates an invariant at a program point by using potentially approximate versions of the execution routes leading thereto, wherein the parts representing the execution of loops have been replaced by approximate steps.

10

19. The method according to claim 18, characterized in that at least a branching declared as "impossible" is proven by analyzing the tree structure routes leading to the transition.

15 20. A method for securing a first program, the first program comprising a finite number of program points and evolution rules associated with the program points and defining the passage from one program point to at least another program point, comprises:

- 20
- a definition of a plurality of exit cases and, when a second program is used in the definition of the first program, for each exit case of the second program, definition of a branching toward a specific program point of the first program or a declaration of branching impossibility;
- 25

- a definition of a set of properties to be proven, each associated with one or more of the constitutive elements of the first program, said set of properties comprising the branching impossibility as a particular property;
- 30

- establishment of the formal proof of the set of properties,

wherein the securing of a first program is based on the securing of a third program having different functionalities

from the first program, and/or not merely being an abstraction of the first program.

21. The method according to claim 20, characterized in that oriented correspondences, called sufficiency relationships, between at least one of the local properties of the first program and at least one of the local properties of the third program are established, such that for proving a local property of the first program, it is sufficient to prove the set of properties of the third program which are in correspondence with this property of the first program, whenever such correspondences exist.

22. The method according to claim 21, characterized in that the third program is modified by a transformation making it possible to obtain the first program, such that the securing of the first program is based on the securing of the third program.

23. The method according to claim 21, characterized in that the first program is modified by a transformation making it possible to obtain the third program, such that the securing of the first program is based on the securing of the third program.

24. The method according to claim 21, characterized in that the first program is modified by a transformation making it possible to obtain the third program and draw up a plurality of sufficiency relationships oriented between at least one of the respective properties of the first and third programs.

25. The method according to claim 21, characterized in that the third program is modified by a transformation making it possible to obtain the first program and to draw up a plurality of sufficiency relationships oriented between at

80

least one of the respective properties of the first and third programs.

CLAIMS

1. A method for securing a first program with a computer system including a processor and memory, the first program comprising a finite number of program points and evolution rules associated with the program points and defining the passage from one program point to another program point, the method comprising:

- defining, with the computer system, a plurality of exit cases in a non-transitory computer readable medium and, when a second program is used in the definition of the first program, the second program defining an executable subroutine of the first program and the second program comprising a finite number of program points and evolution rules associated with the program points and defining a branching toward a specific program point of the first program or a declaration of branching impossibility; and
- defining, with a computer system, a set of local properties to be proven, each associated with one or more evolution rules of the first program, said set of local properties comprising the branching impossibility as a particular local property; and
- establishing, with a computer system, a formal proof of the set of local properties.

2. The method according to claim 1, wherein the evolution rules, the exit cases and the branchings defining a tree structure of execution traces or of logic traces, a prover uses the created tree structure to propagate constraints in points of the first program and uses at least one transitive closure of explicit or implicit congruences in order to prove at least one of the properties of the first program.

3. The method according to claim 2, wherein the first program comprising at least a loop, the prover automatically generates an invariant at a program point by using potentially approximate versions of the execution traces leading thereto, and wherein the execution traces representing the execution of the at least a loop have been replaced by approximate steps.

4. The method according to claim 2, wherein at least a branching declared as "impossible" is proven by analyzing the execution routes or the tree structure of the execution traces leading to the transition.
5. The method according to claim 2, wherein the securing of a first program is based on the securing of a third program having different functionalities from the first program, and/or not merely being an abstraction of the first program .
6. The method according to claim 5, further comprising establishing oriented correspondences, or sufficiency relationships, between a first local property of the first program and a set of local properties of the third program, such that proving the first local property of the first program is sufficient to prove the set of properties of the third program, whenever such correspondences exist.
7. The method according to claim 6, wherein the third program is modified by a transformation making it possible to obtain the first program, such that the securing of the first program is based on the securing of the third program .
8. The method according to claim 6, wherein the first program is modified by a transformation making it possible to obtain the third program, such that the securing of the first program is based on the securing of the third program.
9. The method according to claim 6, wherein the first program is modified by a transformation making it possible to obtain the third program and draw up a plurality of sufficiency relationships oriented between at least one of the respective properties of the first and third programs.
10. The method according to claim 6, wherein the third program is modified by a transformation making it possible to obtain the first program and to draw up a plurality of sufficiency relationships oriented between at least one of the respective properties of the first and third programs.
11. The method according to claim 5, wherein at least one of the variables of the third program is instantiated using values taken by the variables of the first program to draw up at least a sufficiency relationship between the first program and the third program, or as a way of securing the first program.

12. The method according to claim 1, wherein at least one set of local properties is defined by a fourth program.

13. A computer software product comprising software instructions to implement the method according to any one of claims 1 to 12 when said computer software product is executed on a computer.

14. The computer software product according to claim 13, characterized in that it comprises a user interface adapted for enabling a user to define at least one of the plurality of exit cases and associated branchings, and set of local properties to be proven.

15. The computer software product according to claim 14, characterized in that the user interface is adapted to define an impossible branching.

16. The computer software product according to claim 14 or 15, characterized in that the user interface is adapted to define entry conditions of a particular program to use or call at a given point and thus limit the at least one exit case of the called program .

17. The computer software product according to claim 13 that generate implementation code in a way that takes advantage of the formal branch proof of branching impossibility to optimize the generated code.

18. A method for securing a first program, the first program comprising a finite number of program points and evolution rules associated with the program points and defining the passage from one program point to at least another program point, the method comprising:

- defining a plurality of exit cases and, when a second program is used in the definition of the first program, the second program defining an executable subroutine of the first program and the second program comprising a finite number of program points and evolution rules associated with the program points and defining the passage from one program point to another program point until at least one exit case, defining a branching toward a specific program point of the first program or a declaration of branching impossibility; and

- defining a set of local properties to be proven, each associated with one or more program points and evolution rules of the first program, said set of properties comprising the branching impossibility as a particular property;
- establishment of the formal proof of the set of local properties,

wherein the first program comprising at least a loop, a prover automatically generates an invariant at a program point by using potentially approximate versions of the execution routes leading thereto, wherein the execution routes of the at least a loop have been replaced by approximate steps.

19. The method according to claim 18, wherein at least a branching declared as "impossible" is proven by analyzing the execution routes or the tree structure of the execution traces or of logic traces leading to the transition.

20. A method for securing a first program in a non-transitory computer readable medium with a computer system including a processor and memory, the first program comprising a finite number of program points and evolution rules associated with the program points and defining the passage from one program point to at least another program point, comprises:

- defining, with the computer system, a plurality of exit cases in the non-transitory computer medium and, when a second program is used in the definition of the first program, the second program defining an executable subroutine of the first program and the second program comprising a finite number of program points and evolution rules associated with the program points and defining the passage from one program point to another program point until at least one exit case defining a branching toward a specific program point of the first program or a declaration of branching impossibility; and
- defining, with the computer system, a set of local properties to be proven, each associated with one or more program points and evolution rules of the first program, said set of local properties comprising the branching impossibility as a particular local property;
- establishing, with the computer system, a formal proof of the set of properties,

wherein the securing of a first program is based on the securing of a third program having different functionalities from the first program, and/or not merely being an abstraction of the first program.

21. The method according to claim 20, and further comprising oriented correspondences] or sufficiency relationships, between a first local property of the first program and a set of local properties of the third program, such that proving the first local property of the first program is sufficient to prove the set of local properties of the third program, whenever such correspondences exist.

22. The method according to claim 21, wherein the third program is modified by a transformation making it possible to obtain the first program, such that the securing of the first program is based on the securing of the third program.

23. The method according to claim 21, wherein the first program is modified by a transformation making it possible to obtain the third program, such that the securing of the first program is based on the securing of the third program.

24. The method according to claim 21, wherein the first program is modified by a transformation making it possible to obtain the third program and draw up a plurality of sufficiency relationships oriented between at least one of the respective properties of the first and third programs.

25. The method according to claim 21, wherein the third program is modified by a transformation making it possible to obtain the first program and to draw up a plurality of sufficiency relationships oriented between at least one of the respective properties of the first and third programs.