(12) DEMANDE INTERNATIONALE PUBLIÉE EN VERTU DU TRAITÉ DE COOPÉRATION EN MATIÈRE DE BREVETS (PCT)

(19) Organisation Mondiale de la Propriété Intellectuelle
Bureau international

(43) Date de la publication internationale
6 juin 2002 (06.06.2002)

PCT

(10) Numéro de publication internationale
WO 02/44898 A1

(51) Classification internationale des brevets[7] : G06F 9/45

(21) Numéro de la demande internationale :
PCT/FR01/03656

(22) Date de dépôt international :
21 novembre 2001 (21.11.2001)

(25) Langue de dépôt : français

(26) Langue de publication : français

(30) Données relatives à la priorité :
00/15306 24 novembre 2000 (24.11.2000) FR

(71) Déposant *(pour tous les États désignés sauf US)* : **FRANCE TELECOM** [FR/FR]; 6 place d'Allerey, F-75015 PARIS (FR).

(72) Inventeur; et

(75) Inventeur/Déposant *(pour US seulement)* : **BRISSET, Pascal** [FR/FR]; 3, rue Yann Péron, F-22300 Lannion (FR).

(74) Mandataire : **CABINET MARTINET & LAPOUX**; 43, boulevard Vauban, Guyancourt, Boîte postale 405, F-78055 Saint Quentin Yvelines Cedex (FR).

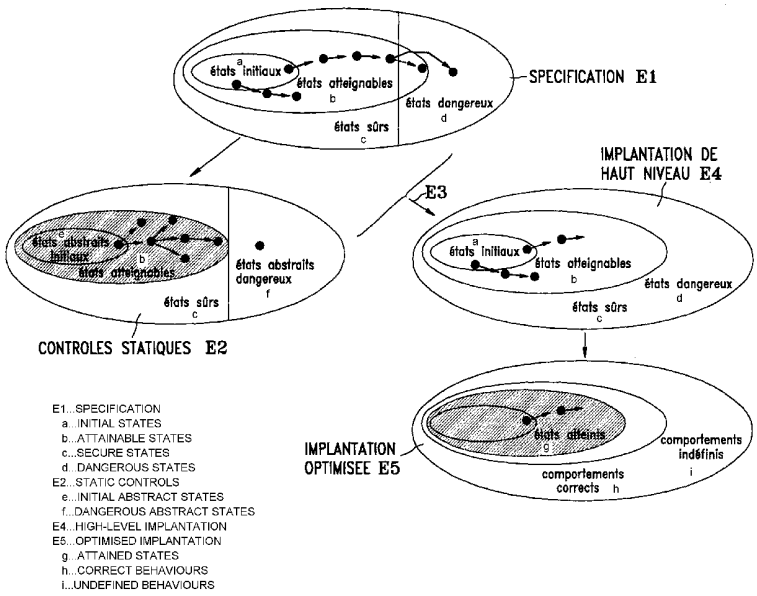
(81) États désignés *(national)* : AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) États désignés *(régional)* : brevet ARIPO (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), brevet

[Suite sur la page suivante]

(54) Title: FORMAL VERIFICATION IN PARTICULAR OF A SECURED VIRTUAL MACHINE

(54) Titre : VERIFICATION FORMELLE NOTAMMENT D'UNE MACHINE VIRTUELLE SECURISEE



(57) **Abstract:** The invention concerns formal verification and optimisation, typically of a virtual machine initially written in high-level language and implanted for example in a smart card. During verification, it is formally proved (E4) that controls over explored states of the programme (E1-E2) by security mechanisms guarantee that a specific forbidden state in high-level language is uttainable by the programmes. The implantation of the programme is then optimised in particular by eliminating execution paths leading to the forbidden state in the programme, so as to transform it into a low-level language providing the same security guarantees as the high-level language programme.

[Suite sur la page suivante]

WO 02/44898 A1

eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Déclarations en vertu de la règle 4.17 :

— *relative au droit du déposant de revendiquer la priorité de la demande antérieure (règle 4.17.iii)) pour la désignation suivante US*

— *relative à la qualité d'inventeur (règle 4.17.iv)) pour US seulement*

Publiée :

— *avec rapport de recherche internationale*

En ce qui concerne les codes à deux lettres et autres abréviations, se référer aux "Notes explicatives relatives aux codes et abréviations" figurant au début de chaque numéro ordinaire de la Gazette du PCT.

(57) Abrégé : L'invention concerne la vérification formelle et l'optimisation d'un programme, typiquement une machine virtuelle, initialement écrit dans un langage de haut niveau et implanté par exemple dans une carte à puce. Au cours de la vérification, il est prouvé formellement (E4) que des contrôles sur des états du programme explorés (E1-E2) par des mécanismes de sécurité garantissent qu'un état interdit défini en langage de haut niveau n'est pas atteignable par les programme. L'implantation du programme est alors optimisée notamment en supprimant des chemins d'exécution conduisant à l'état interdit dans le programme, de manière à le transformer en un programme en langage de bas niveau fournissant les mêmes garanties de sécurité que le programme en langage de haut niveau.

Vérification formelle notamment d'une machine virtuelle sécurisée

La présente invention concerne la vérification d'un programme initialement écrit en un langage de haut niveau, lors de sa réalisation dans un milieu sécurisé et ainsi liée à des mécanismes de sécurité contrôlant des états possibles du programme.

Par exemple, le programme est un interpréteur ou une machine virtuelle implantée dans une carte à puce ou un terminal radiotéléphonique portable.

Dans les applications globales liées au réseau Internet, les éditeurs d'outils informatiques, particulièrement de navigateurs, ont été contraints à adopter un langage commun de haut niveau, tel que le langage orienté objet, appelé Java (marque déposée), pour la programmation répartie de communications entre des petits programmes locaux et des serveurs.

La recherche de fonctionnalités, notamment de sécurité, de plus en plus nombreuses et souples pour des objets nomades, tels que radiotéléphones mobiles, cartes de paiement et assistants numériques personnels, a conduit à doter les microcontrôleurs inclus dans des cartes à puce et moyens de traitement de données analogues, de langages relativement complets tels que le langage Java.

L'universalité, dans la variété des dispositifs connectés au plus grand des réseaux comme dans celles d'appareils de plus en plus petits, issus d'une multiplicité innombrable de constructeurs, d'architectures matérielles différentes, de systèmes informatiques divers, dans des contraintes très diverses, est un obstacle à un langage unique, interprété sans ambiguïté.

Pour ces raisons a été définie une machine virtuelle capable d'exécuter tous les programmes écrits dans le langage Java. Les fournisseurs de matériel notamment de cartes à puce, ou les éditeurs de logiciels notamment de navigateurs Web, ont dû alors développer sur les outils qu'ils fournissent, un logiciel capable de réaliser les fonctions de cette machine virtuelle, appelée habituellement "machine virtuelle Java" JVM, propre à chaque outil logiciel ou matériel. A cause de la faible taille de la mémoire dans les cartes à puce, ce logiciel alors appelé "JavaCard" a été "allégé".

Contrairement aux processeurs "nus", dont l'objectif est avant tout la performance de calcul ou de faible consommation, la machine virtuelle Java a été conçue pour fournir aux développeurs des fonctions de sécurité adaptées à des utilisations sensibles notamment dans le domaine de la monétique et de la sécurité.

Par contre, l'exécution d'un programme Java n'est effectivement totalement sûre que si la machine JVM a été correctement réalisée pour toutes les fonctions critiques de sécurité.

Les normes ITSEC (Information Technology Security Evaluation Criteria) de la Commission des Communautés Européennes conseillent pour l'analyse de la sécurité de systèmes informatiques :

- de fixer des objectifs de sécurité ;
- d'en déduire une politique de sécurité dont l'application permettra d'atteindre les objectifs de sécurité ;
- de définir des fonctions de sécurité dont l'exécution garantit que la politique de sécurité est respectée ;

- de concevoir des mécanismes de sécurité qui sont la réalisation matérielle ou logicielle des fonctions de sécurité.

Il est donc important pour les clients-utilisateurs finaux que la machine JVM soit conforme à la politique de sécurité que la nature de l'application par exemple dans le domaine de la monétique ou de la radiotéléphonie mobile, impose aux opérateurs, tels que banques ou opérateurs de télécommunication.

Dans ce contexte, un fournisseur de la machine JVM supportée par un moyen de traitement de données a intérêt à démontrer que sa réalisation est conforme à une politique de sécurité que son client, opérateur ou banque, lui aura contractuellement définie, ou à celle que la loi ou les règlements imposent.

Il faudra donc vérifier que telle ou telle faille de sécurité n'existe pas, quel que soit le programme Java exécuté par la machine JVM, et quel que soit l'environnement du moyen de traitement de données, tel qu'un processeur, qui exécute la machine JVM. Il s'agit donc d'un processus complexe et délicat, avec des implications économiques fortes.

Une telle vérification repose sur des techniques formelles qui sont un ensemble d'outils, logiciels ou méthodologiques, garantissant de manière certaine des propriétés de logiciels. Au cours du processus de développement d'un programme, par exemple un interpréteur, en l'occurrence une machine virtuelle, ces techniques formelles mettent en oeuvre des démarches mathématiques qui assurent ces garanties. De nombreuses techniques sont disponibles, chacune ayant ses spécificités.

Pour des raisons d'efficacité, certains des contrôles effectués par la machine virtuelle sont statiques, comme l'analyse sémantique d'un programme avant son exécution. Comme les algorithmes mis en jeu sont complexes, il est difficile de concevoir et d'implanter, c'est-à-dire réaliser une machine virtuelle. La vérification qu'une machine virtuelle Java respecte bien une politique de sécurité recherchée nécessite de passer, pour certaines propriétés, par l'utilisation conjointe de plusieurs techniques, formalismes, langages, qui structurent le développement.

Plus particulièrement, l'invention concerne un procédé de vérification qui garantit que la spécification d'une machine virtuelle est correcte et non ambiguë, et que son implantation est sûre. Il s'agit de vérifier formellement la correction des contrôles statiques et de leur implantation.

L'invention a pour objectif d'optimiser l'implantation d'un interpréteur de programmes en langage de haut niveau, tel qu'une machine virtuelle dont les mécanismes de sécurité, comportant par exemple des contrôles statiques, ont été vérifiés formellement en conformité avec une spécification de fonction de sécurité.

A cette fin, un procédé pour vérifier et optimiser un programme initialement écrit en un langage de haut niveau et implanté dans un moyen de traitement de données, au cours duquel des contrôles sur des états du programme explorés par des mécanismes de sécurité prouvent formellement qu'un état interdit défini en langage de haut niveau est inatteignable par le programme, est caractérisé

par une suppression de chemins d'exécution conduisant à l'état interdit dans le programme, de manière à transformer le programme en un programme équivalent écrit dans un langage de bas niveau. Ce dernier fournit alors les mêmes garanties de sécurité que le programme en langage de haut niveau.

L'implantation en langage de haut niveau est ainsi transformée en une implantation optimisée dans un langage de bas niveau par application manuelle ou automatique de règles de transformation locales sur le code source de haut niveau. La simplicité et le caractère systématique de ces règles garantit semi-formellement ou formellement la correction de l'implantation optimisée en langage de bas niveau par rapport à l'implantation de haut niveau, et par transitivité, par rapport à des spécifications des mécanismes de sécurité.

Selon d'autres caractéristiques de l'invention, l'optimisation du programme, tel qu'interpréteur ou machine virtuelle, peut comprendre, en outre, un remplacement d'entiers non bornés du langage de haut niveau, par des entiers bornés du langage de bas niveau et/ou un remplacement de paramètres et d'appels de fonction en langage de haut niveau par des données allouées statiquement et des structures de contrôles impératives en langage de bas niveau.

Le procédé de vérification selon l'invention peut être appliqué à un programme du type machine virtuelle connue comportant des données entières, des tableaux, des pointeurs sur les tableaux, des variables locales réutilisables ou registres, des exceptions, des sous-routines, ou une pile d'opérandes. Le jeu d'instructions de la machine virtuelle comporte des opérations arithmétiques,

d'accès aux variables, d'accès aux tableaux, de manipulation de la pile, de test, de saut, d'appel et retour de sous-routines, de levée d'exceptions.

Les contrôles statiques garantissent le respect de contraintes de typage des opérandes, de contraintes sur le flot de contrôle, de contraintes de non-débordement de la pile d'opérandes, et de contraintes sur l'utilisation de variables locales.

D'autres caractéristiques et avantages de la présente invention apparaîtront plus clairement à la lecture de la description suivante de plusieurs réalisations préférées de l'invention en référence aux dessins annexés correspondants dans lesquels :

- la figure 1 est un algorithme d'interpréteur en langage de haut niveau avec contrôles dynamiques ;
- la figure 2 est un algorithme optimisé d'interpréteur en langage de bas niveau ; et
- la figure 3 illustre schématiquement un procédé de vérification formelle de machine virtuelle aboutissant à une optimisation selon l'invention.

A titre d'exemple, on se réfère à un programme de type interpréteur constituant le moteur d'exécution d'une machine virtuelle implantée dans un moyen de traitement de données, dite plate-forme d'exécution, tel que le microcontrôleur d'un terminal radiotéléphonique mobile, ou d'une carte à puce telle qu'une carte de paiement ou une carte d'identité SIM (Subscriber Identity Module). L'interpréteur implémenté automatiquement à partir de spécifications formelles et écrit en langage source de haut niveau pour exécuter une instruction comme montré à la figure 1, est à optimiser selon l'invention en un langage de bas niveau montré à la figure 2. Par

exemple, le langage source de haut niveau est un langage de la famille ML, tel que le langage CAML développé par l'INRIA en France, et le langage de bas niveau est le langage impératif C.

L'implantation de l'interpréteur (interp) en langage de haut niveau est obtenue par un procédé automatique à partir de spécifications formelles écrites dans un langage à base de logique mathématique, ce qui assure sa conformité à ces spécifications. Elle comporte la structure de contrôle suivante :

```
let interp m st =
 match (nth st.pc m.code) with
|None->Interdit
|Some Illegal->Interdit
|Some Iadd->
   (match st.stack with
   |Cons(Vint   x,Cons(Vint   y,stack'))->Continuer
[...]
   |_-> Interdit
[...]
```

Cette structure de contrôle assure les fonctions suivantes en référence aux étapes H1 à H7 de la figure 1.

A l'étape H1, lors d'une tentative de lecture de l'instruction courante I désignée par le compteur ordinal (st.pc) dans un état (st) pour un programme (m.code), la valeur de l'adresse d'exécution correspondant à l'instruction courante est contrôlée (match (nth st.pc m.code)). Si l'adresse est invalidée puisqu'elle n'appartient pas au programme (None) ou désigne une instruction incorrecte (Some

Illegal), le contrôle passe à un état "interdit" à l'étape H7 où il est arrêté. Sinon, à l'étape suivante H2, l'instruction courante I pointée par l'adresse d'exécution validée est contrôlée.

Par exemple, l'instruction validée à l'étape H2 peut être l'instruction d'addition (Iadd). Dans ce cas, aux étapes suivantes H3 et H4 qui peuvent être réunies, les opérandes auxquels l'instruction d'addition est appliquée sont contrôlés. En l'occurrence, si le sommet de la pile d'opérandes contient au moins deux valeurs (Cons valeur1 (Cons valeur2 stack')) et que ces deux valeurs sont de type entier (Vint), c'est-à-dire si l'addition est cohérente, l'exécution se poursuit normalement (Continuer), en dépilant les valeurs à l'étape H5, en empilant leur somme à l'étape H6, et en incrémentant le compteur ordinal et appelant récursivement l'interpréteur (interp m st') sur un nouvel état (st') de manière à revenir à l'étape H1. Par contre, si l'instruction courante est Iadd alors que les opérandes ne sont pas en nombre suffisant ou ne sont pas du bon type, le contrôle passe à l'état interdit et est arrêté à l'étape H7.

Selon une autre variante également montrée à la figure 1, lorsque l'instruction courante est l'instruction d'empilement d'une valeur constante C, la structure de contrôle comporte les étapes H8 et H9. L'étape H8 contrôle la hauteur de la pile d'opérandes et, si la pile n'est pas pleine, l'étape H9 empile la valeur C au sommet de la pile. Par contre, si la pile est pleine, la structure de contrôle va à l'état interdit à l'étape H7.

Dans la figure 1, la structure de contrôle comporte trois chemins d'exécution issus des étapes H1, H2 et (H3,H4), ou bien des étapes H1, H2 et H8,

qui correspondent à des échecs de contrôles et qui aboutissent à l'état interdit à l'étape H7.

En référence maintenant à la figure 2, l'interpréteur en langage de bas niveau C après optimisation selon l'invention appliquée à l'interpréteur en langage de haut niveau ML, ne comporte plus que des étapes de processus B1, B2, B5, B6 et B9 correspondant respectivement aux étapes H1, H2, H5, H6 et H9, sans les étapes de contrôle dynamique H3-H4 et H8 et surtout sans l'étape d'état interdit H7.

La structure de contrôle optimisée en langage de bas niveau C s'écrit :

```
switch (code[pc]){
   case IADD:stack[top+1]+=stack[top];++top;++pc;break;
   [...]
}
```

où l'instruction d'analyse conditionnelle (switch) est lue à l'étape B1 pour décoder l'instruction suivante (case) associée à la valeur (pc) à l'étape B2 et désignant l'instruction d'addition (IADD) de deux valeurs entières d'adresses (top+1) et (top) à dépiler à l'étape B5.

Ce code source de bas niveau est à la fois performant puisqu'il ne comporte pas de contrôles dynamiques inutiles et est sûr puisqu'il est directement dérivé d'un code source dont la correction a été prouvée formellement selon l'invention, comme on le verra ci-après.

Comparativement à la figure 1, les chemins d'exécution aboutissant à l'état interdit de l'étape H7 sont supprimés du code source de l'interpréteur

dans la figure 2, ce qui correspond aux optimisations suivantes :

- suppression de contrôle sur l'adresse d'exécution à l'étape H1 correspondant à l'étape B1 ;
- suppression du contrôle et du type des arguments auxquels l'opération d'addition est appliquée aux étapes H3 et H4 ;
- simplification de la représentation des données en machine, leur type n'étant plus représenté.

Ces optimisations appliquées au code source de haut niveau ci-dessus ML conduisent, par l'application de transformations simples et locales, soit manuellement soit par l'utilisation d'un outil automatique, à un code source optimisé dans un langage de bas niveau, en l'occurrence le langage C.

Le procédé de l'invention comporte l'obtention d'une preuve formelle de l'efficacité des mécanismes de contrôle statique de la machine virtuelle. En d'autres termes, si ces mécanismes ont autorisé l'exécution d'un programme donné (code), alors l'exécution de ce programme par l'interpréteur (interp) n'aboutira jamais à l'état interdit. Cette garantie est obtenue au cours des étapes suivantes E1 à E5 montrées à la figure 3.

A l'étape E1, un langage logique de spécification de la machine virtuelle qui est une variante de la théorie des types, permet de décrire et de raisonner sur des structures de données et des algorithmes dans un programme. Des mécanismes de sécurité prédéterminés, tels que des contrôles statiques (instruction ou adresse d'exécution incorrecte ; étape H1 ou H2), sont spécifiés comme un

problème d'analyse de flot des états, ou variantes, possibles d'exécution du programme réalisant l'interpréteur, d'une manière analogue à l'analyse des comportements d'un objet symbolique. Des fonctions de sécurité comportent typiquement des contrôles de typage, d'accès aux données, d'accès aux opérations, et d'accès aux ressources. Si certains états atteignables par l'exécution du programme depuis des états initiaux de ceux-ci deviennent dangereux, ces états sont ramenés par des transitions à un état interdit (étape H7), et les autres états sont réputés sûrs.

A l'étape E2, parmi les mécanismes de sécurité prédéterminés, des mécanismes de sécurité sont obtenus par reformulation du problème d'analyse de flot en la combinaison d'un problème d'abstraction et d'un problème d'exploration exhaustive d'un système à états et transitions. S'il existe un nombre infini d'états d'exécution du programme, ce nombre infini est ramené à un nombre fini d'états abstraits atteignables du programme qui sont explorés par les contrôles statiques. Les contrôles statiques vérifient que l'état abstrait "interdit" est inatteignable par le programme ainsi défini de manière à préserver des propriétés de sécurité du programme.

L'étape E3 consiste à passer des étapes E1 et E2 à une étape E4, c'est-à-dire à spécifier l'interpréteur de la machine virtuelle pour qu'il comporte des assertions, par exemple les validations d'adresses ou d'instructions aux étapes H1, H2 et les contrôles à l'étape de contrôle d'opérandes entiers H3-H4. Ces assertions expriment une politique de sécurité, ainsi qu'un état dit interdit (étape H7) qui est atteint chaque fois qu'une assertion échoue.

L'interpréteur et ses mécanismes de sécurité sont implantés en langage de haut niveau de type ML, par exemple le langage CAML selon l'exemple ci-dessus et la figure 1, ou le langage SML, à partir de spécifications de la machine virtuelle et à l'aide d'un outil à base de logique. Cette implantation à l'étape E4 comporte des contrôles dynamiques correspondant aux assertions, lesquels contrôles dynamiques ramènent les états dangereux du programme à l'état interdit. Au cours de l'étape E4, il est prouvé à l'aide de l'outil à base de logique que si les mécanismes ont autorisé l'exécution d'un programme, alors son exécution par l'interpréteur n'aboutira jamais à l'état interdit.

Puis selon l'invention, l'étape E5 optimise l'interpréteur de la machine virtuelle par application de trois transformations locales suivantes sur le code source en langage ML. Ces transformations sont réalisées manuellement par un programmeur, bien qu'en variante au moins certaines d'entre elles peuvent être réalisées automatiquement par des outils de programmation appropriés.

E51) Une première transformation est une suppression des chemins d'exécution, par exemple entre les étapes H1, H2, H3, H4 et l'étape H7, qui conduisent de façon certaine à l'état interdit. Cette suppression est garantie par les contrôles statiques qui ont assuré qu'aucun état atteignable par l'interpréteur n'est dangereux. En outre, les contrôles statiques justifient la simplification de la représentation des données en machine, la suppression de tests de débordement d'indices, et la suppression de tests sur le compteur ordinal d'instructions, comme montré par la comparaison des figures 1 et 2.

E52) Une deuxième transformation est un remplacement des types entiers infinis, c'est-à-dire non bornés, du langage de haut niveau par des types entiers bornés, c'est-à-dire des entiers binaires finis, dans le langage de bas niveau C. Ce remplacement est effectué selon une première variante, lorsqu'il a été prouvé formellement que des bornes prédéterminées ne peuvent pas être atteintes par des variables entières du langage de haut niveau traitées par l'interpréteur. Selon une deuxième variante, ce remplacement est effectué lorsque les opérations appliquées sur ces variables entières sont telles que les bornes ne peuvent pas être atteintes par les variables entières en langage de haut niveau avant l'expiration d'une durée prédéterminée inférieure à la durée de vie de la machine virtuelle ; par exemple cette deuxième variante est appliquée lorsque les seules opérations sont des incrémentations et des décrémentations relatives à une valeur initiale faible.

E53) Une troisième transformation est un remplacement des appels de fonction dits "récursifs terminaux" (en anglais "tail-recursive") et de leurs arguments en langage de haut niveau par des structures de contrôles impératives en langage de bas niveau et des données allouées statiquement. Par exemple, les appels récursifs terminaux de l'interpréteur (interp) sont remplacés par une boucle impérative, et son argument (st) représentant l'état du programme est remplacé par des données allouées statiquement, dont entre autres la pile d'opérandes (stack) et le compteur ordinal (pc) dans l'exemple en langage de bas niveau.

Le domaine applicatif du procédé de vérification de l'invention concerne notamment des cartes à puce monétiques ou pour accès sécuritaire, et tout particulièrement des cartes à puce téléchargeables dont le code exécuté n'est pas connu a priori. Les cartes à puce peuvent être incluses dans des dispositifs dont la programmation est accessible à des parties tierces, notamment pour des applications de radiotéléphonie mobile, et tout particulièrement des applications téléphoniques Wap mêlant les deux aspects internet et mobile.

REVENDICATIONS

1 - Procédé pour vérifier et optimiser un programme initialement écrit en un langage de haut niveau et implanté dans un moyen de traitement de données, au cours duquel des contrôles (H1, H2, H3-H4) sur des états du programme explorés (E1-E2) par des mécanismes de sécurité prouvent formellement (E4) qu'un état interdit (H7) défini en langage de haut niveau est inatteignable par le programme, caractérisé par une suppression de chemins d'exécution (H1-H7, H2-H7, H3-H4-H7) conduisant à l'état interdit dans le programme, de manière à transformer le programme en un programme équivalent dans un langage de bas niveau.

2 - Procédé conforme à la revendication 1, comprenant un remplacement d'entiers non bornés du langage de haut niveau, par des entiers bornés du langage de bas niveau.

3 - Procédé conforme à la revendication 2, selon lequel ledit remplacement est effectué lorsqu'il a été prouvé formellement que des bornes prédéterminées des entiers du langage de haut niveau ne peuvent pas être atteintes.

4 - Procédé conforme à la revendication 2, selon lequel ledit remplacement est effectué lorsque des bornes prédéterminées ne peuvent pas être atteintes par les entiers en langage de haut niveau avant l'expiration d'une durée prédéterminée.

5 - Procédé conforme à l'une quelconque des revendications 1 à 4, comprenant un remplacement de

paramètres et d'appels de fonction en langage de haut niveau par des données allouées statiquement et des structures de contrôles impératives en langage de bas niveau.

FIG. 2

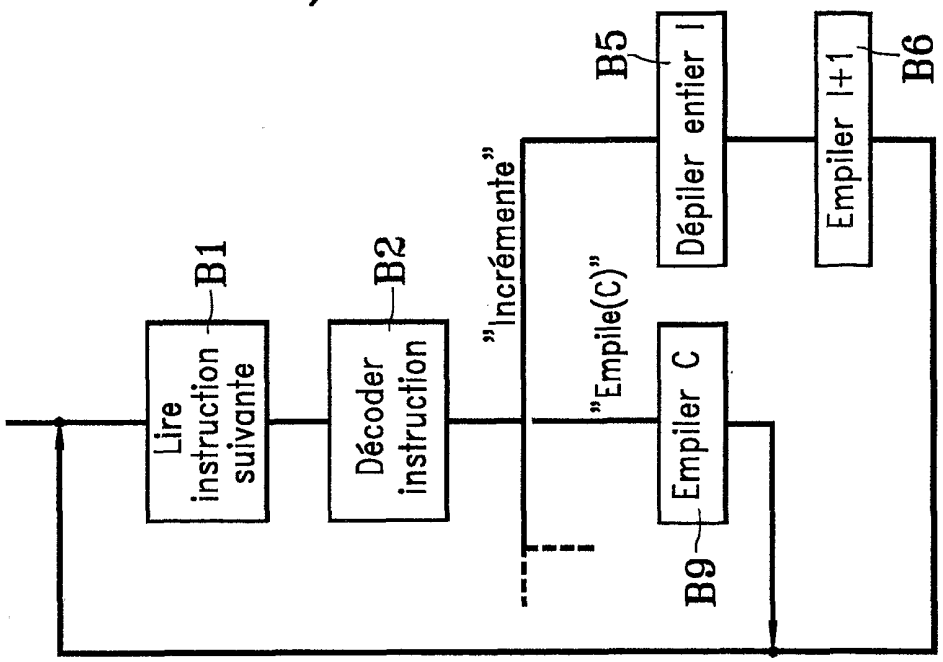
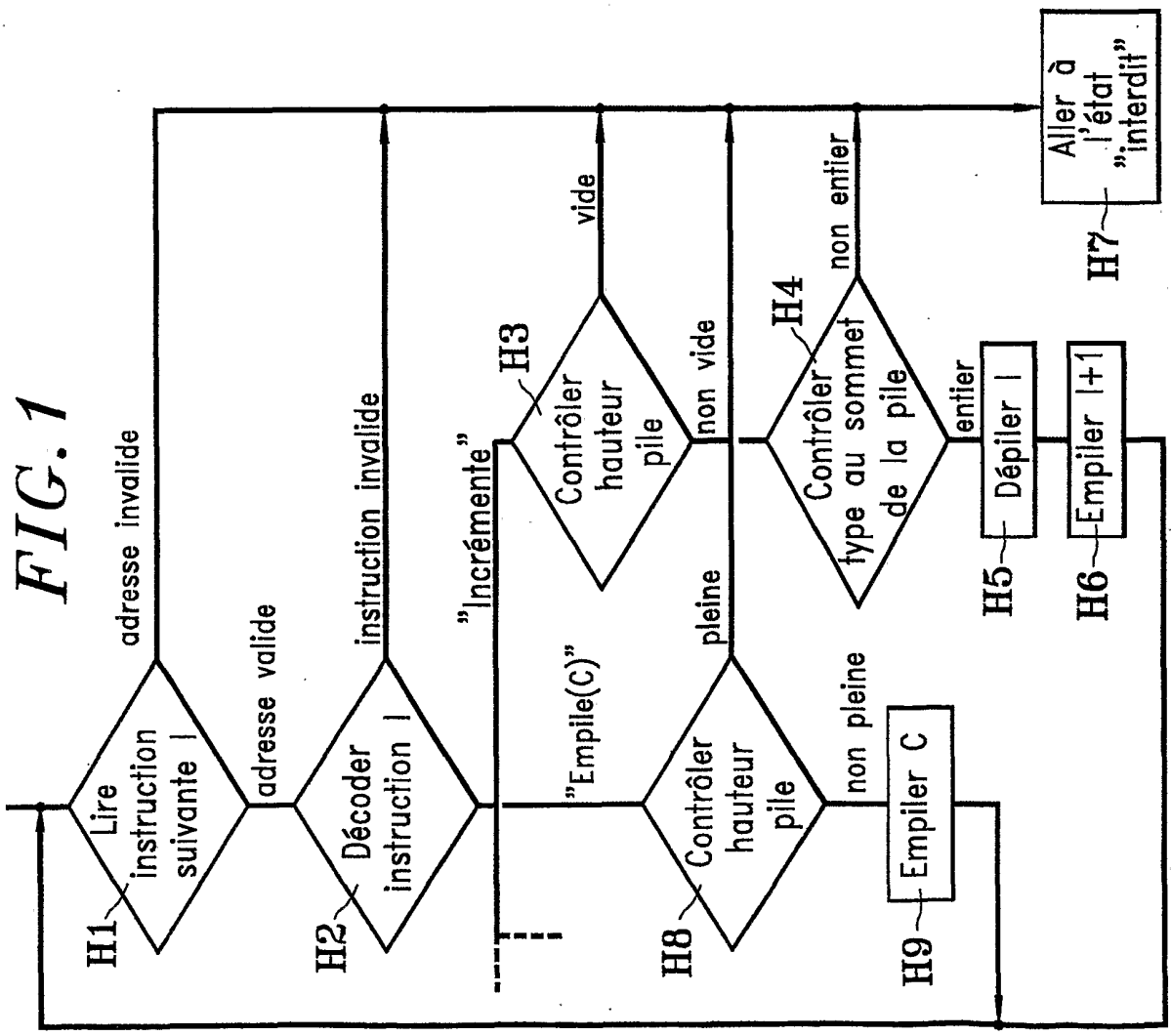

Lire instruction suivante
B1
Décoder instruction
B2
"Incrémente"
"Empile(C)"
Empiler C
B9
Dépiler entier I
B5
Empiler I+1
B6

FIG. 1


Lire instruction suivante I
H1
adresse invalide
adresse valide
Décoder instruction I
H2
instruction invalide
"Incrémente"
"Empile(C)"
Contrôler hauteur pile
H3
vide
non vide
Contrôler type au sommet de la pile
H4
non entier
entier
Aller à l'état "interdit"
H7
Dépiler I
H5
Empiler I+1
H6
Contrôler hauteur pile
H8
pleine
non pleine
Empiler C
H9

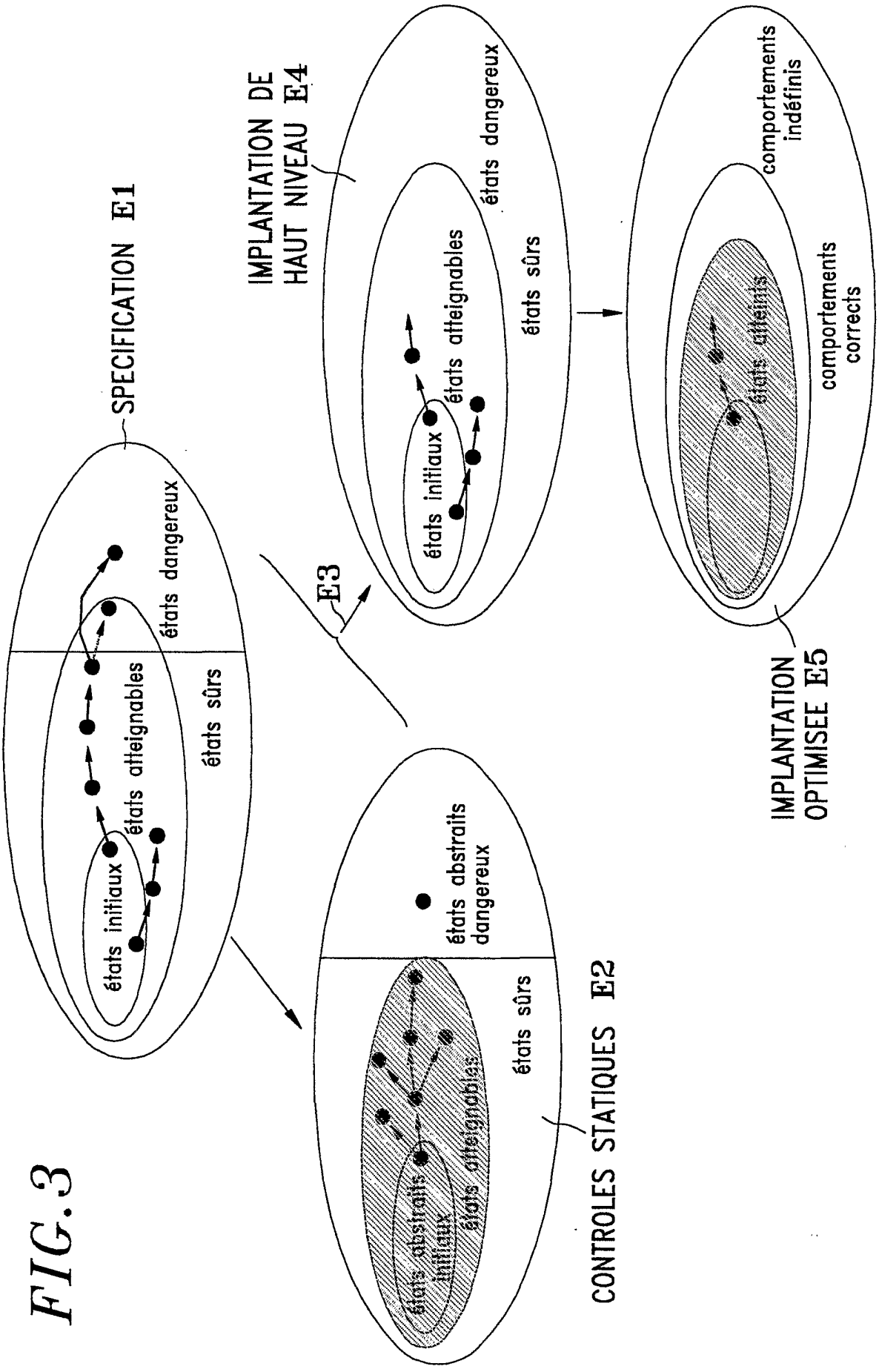

SPECIFICATION E1
états initiaux
états atteignables
états dangereux
états sûrs
E3
IMPLANTATION DE HAUT NIVEAU E4
états initiaux
états atteignables
états dangereux
états sûrs
CONTROLES STATIQUES E2
états abstraits initiaux
états abstraits atteignables
états abstraits dangereux
états sûrs
IMPLANTATION OPTIMISEE E5
états atteints
comportements corrects
comportements indéfinis

FIG.3

INTERNATIONAL SEARCH REPORT

In tional Application No
PCT/FR 01/03656

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/45

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	AMMONS G ET AL: "IMPROVING DATA-FLOW ANALYSIS WITH PATH PROFILES" ACM SIGPLAN NOTICES, ASSOCIATION FOR COMPUTING MACHINERY, NEW YORK, US, vol. 33, no. 5, 1 May 1998 (1998-05-01), pages 72-84, XP000766261 ISSN: 0362-1340 the whole document	1
A	US 5 724 590 A (KRANTZLER IRVAN JAY ET AL) 3 March 1998 (1998-03-03) column 20, line 38 -column 27, line 30	1

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
22 February 2002	01/03/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Bijn, K

INTERNATIONAL SEARCH REPORT

Information on patent family members

In tional Application No
PCT/FR 01/03656

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5724590 A	03-03-1998	AU 613248 B2	25-07-1991
		AU 4474189 A	02-08-1990
		CA 2002201 A1	06-06-1990
		DE 68926706 D1	25-07-1996
		DE 68926706 T2	09-01-1997
		EP 0372834 A2	13-06-1990
		HK 179496 A	04-10-1996
		JP 2010818 C	02-02-1996
		JP 2238527 A	20-09-1990
		JP 7031604 B	10-04-1995

RAPPORT DE RECHERCHE INTERNATIONALE

D ıde Internationale No
PCT/FR 01/03656

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
CIB 7 G06F9/45

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
CIB 7 G06F

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si réalisable, termes de recherche utilisés)
EPO-Internal

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie °	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
A	AMMONS G ET AL: "IMPROVING DATA-FLOW ANALYSIS WITH PATH PROFILES" ACM SIGPLAN NOTICES, ASSOCIATION FOR COMPUTING MACHINERY, NEW YORK, US, vol. 33, no. 5, 1 mai 1998 (1998-05-01), pages 72-84, XP000766261 ISSN: 0362-1340 le document en entier	1
A	US 5 724 590 A (KRANTZLER IRVAN JAY ET AL) 3 mars 1998 (1998-03-03) colonne 20, ligne 38 -colonne 27, ligne 30	1

☐ Voir la suite du cadre C pour la fin de la liste des documents

☒ Les documents de familles de brevets sont indiqués en annexe

° Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent

"E" document antérieur, mais publié à la date de dépôt international ou après cette date

"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)

"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens

"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée

"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention

"X" document particulièrement pertinent; l'inven tion revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément

"Y" document particulièrement pertinent; l'inven tion revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier

"&" document qui fait partie de la même famille de brevets

Date à laquelle la recherche internationale a été effectivement achevée	Date d'expédition du présent rapport de recherche internationale
22 février 2002	01/03/2002

Nom et adresse postale de l'administration chargée de la recherche internationale	Fonctionnaire autorisé
Office Européen des Brevets, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Bijn, K

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux membres de familles de brevets

Demande Internationale No
PCT/FR 01/03656

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
US 5724590 A	03-03-1998	AU 613248 B2	25-07-1991
		AU 4474189 A	02-08-1990
		CA 2002201 A1	06-06-1990
		DE 68926706 D1	25-07-1996
		DE 68926706 T2	09-01-1997
		EP 0372834 A2	13-06-1990
		HK 179496 A	04-10-1996
		JP 2010818 C	02-02-1996
		JP 2238527 A	20-09-1990
		JP 7031604 B	10-04-1995