I416412

發明專利說明書

（本說明書格式、順序及粗體字，請勿任意更動，※記號部分請勿填寫）

※ 申請案號：93112542

※ 申請日期：93 年 5 月 4 日　　※IPC 分類：G06F 9/45 (2006.01)

一、發明名稱：（中文/英文）

在編譯處理的期間表示及檢測程式元件的一致性之可擴充式系統、電腦可讀取媒體、電腦程式產品及方法

EXTENSIBLE TYPE SYSTEM, COMPUTER-READABLE MEDIUM, COMPUTER PROGRAM PRODUCT AND METHOD FOR REPRESENTING AND CHECKING CONSISTENCY OF PROGRAM COMPONENTS DURING THE PROCESS OF COMPILATION

二、申 請 人：（共 1 人）

姓名或名稱：（中文/英文）

美商．微軟公司

Microsoft Corporation

代表人：（中文/英文）

艾苹那諾爾 D 巴特萊

EPPENAUER, D. BARTLEY

住居所或營業所地址：（中文/英文）

美國華盛頓州列德蒙微軟路 1 號

One Microsoft Way, Building 8, Redmond, WA 98052-6399, U.S.A.

國　籍：（中文/英文）

美國/USA

三、發明人：（共 2 人）

姓　名：（中文/英文）

1.珀雷斯科馬克羅奈爾德/PLESKO, MARK RONALD

2.塔迪堤大衛瑞德二世/TARDITI, DAVID READ, JR.

國　籍：(中文/英文)

1.美國/USA

2.美國/USA

四、聲明事項：

☐ 主張專利法第二十二條第二項☐第一款或☐第二款規定之事實，其事實發生日期為：　年　月　日。

☑ 申請前已向下列國家（地區）申請專利：

【格式請依：受理國家（地區）、申請日、申請案號　順序註記】

☑ 有主張專利法第二十七條第一項國際優先權：

美國；2003 年 6 月 27 日；10/607,601

☐ 無主張專利法第二十七條第一項國際優先權：

☐ 主張專利法第二十九條第一項國內優先權：

【格式請依：申請日、申請案號　順序註記】

☐ 主張專利法第三十條生物材料：

☐ 須寄存生物材料者：

國內生物材料　【格式請依：寄存機構、日期、號碼　順序註記】

國外生物材料　【格式請依：寄存國家、機構、日期、號碼　順序註記】

☐ 不須寄存生物材料者：

所屬技術領域中具有通常知識者易於獲得時，不須寄存。

五、中文發明摘要：

本發明提供一種用以檢查在各式中介語言內之一致性的型態表現、型態檢查器及編譯器。可藉由採用一或更多的規則集合作為對一型態檢查器之輸入，這會按照各式關鍵項之任一者，或是該等二者以上之組合，來選擇一或更多規則集合，以在一編譯器內進行程式語言的型態檢查作業。在這些之中，會是編譯階段、來源語言、架構及定態層級出現在所進行型態檢查之語言內。然後會利用經選擇的一或更多規則集合，對該語言進行型態檢查。該等規則集合包括一對應於強型態檢查之規則集合、一對應於弱型態檢查之規則集合，及一對應於表現型態檢查之規則集合。在替代方式裡，一編譯器可提供一型態檢查器，此者可根據先前說明之關鍵項的其中一者或是兩者以上之組合，從一較大的規則集合裡，在執行時間內建構出一或更多的規則集合。

六、英文發明摘要：

A representation of types, type-checker and compiler are provided fro checking consistency in various forms of an intermediate language. Type-checking a programming language in compiler is accomplished by taking one or more rule sets as input to a type-checker, which selects one or more of the rules sets based upon any one, or combination of two or more, of numerous criteria. Among them are stage of compilation, source language, architecture, and level of typing present in the language being type-checked. The language is then type-checked using the selected one or more rule sets. The rule set can include one rule set corresponding to strong type-checking, one rule set corresponding to weak type-checking, and one rule set corresponding to representation type-checking. In the alternative, a compiler can be provided with a type-checker that constructs the one or more sets of rules at runtime from a larger set of rules based on any one, or combination of two or more, of the previously mentioned criteria.

101年1月11日修正替換頁

七、指定代表圖：

(一)、本案指定代表圖為：第 3 圖。

(二)、本代表圖之元件代表符號簡單說明：

300 來源程式碼

302 HIR

304 MIR

306 LIR

308 型態檢查器

310 規則集合

八、本案若有化學式時，請揭示最能顯示發明特徵的化學式：

無

九、發明說明：

【發明所屬之技術領域】

本發明特點係有關於型態系統，尤指關於一種可擴充至新近及經更新之程式語言的型態系統。

【先前技術】

型態系統是一種用於程式語言以輔助偵測及防止執行時間錯誤的系統。一程式語言如果含有一組被宣告為像是變數、函式等之物件的型態，並且會在程式編譯的過程中基於一組規則集合來檢查這些型態，則此者屬「型態式」。如按該型態式語言所撰寫之來源程式碼違反各型態規則的其中一項，則會決定為一編譯器錯誤。

在過去多年裡，運用於編譯器的型態中介語言在研究社群中獲得廣泛研究。該等增強了編譯器的可靠性及強固性，而且提供一種系統性的方式來追蹤並檢查垃圾回收器的所需資訊。此構想係具有一中介表現，此者具有接附其上之型態，且可按類比於對來源程式碼程式進行型態檢查之方式來進行型態檢查。然而，型態式中介語言會更難以實作，這是因為在編譯過程中，代表被令屬顯明之項目的型態會是有必要的。

如必須要表現數種不同高階程式語言，則型態式中介語言會甚更為難以實作。不同語言不僅會具有相異的原生運算及型態，同時高階程式語言會具有不同的定態層級。例如，有些像是組合語言的語言一般會屬於非型態式者。

換言之，該等並無型態系統。在屬型態式的語言中，有些會是強型態式，而有些會是較為鬆散之型態式。例如，C++一般被視為是鬆散型態式的語言，而ML或PASCAL則是被視為是強型態式語言。此外，有些屬鬆散型態式的語言會具有該語言的一些較小子集合，該等允許一程式內的多數程式碼區段為強型態式，而其他程式碼區段則為鬆散型態式。例如C#及用於.NET內的微軟「中介語言」(MSIL)允許此者。因此，一用以表現這些高階語言任者的型態式中介語言必須能夠表現不同的型態強度。類似地，該型態式中介語言的型態系統必須要能夠按照所受型態檢查之程式碼的特徵，以實作出不同的規則。

試圖產生型態式中介語言通常會無法解決上述問題。例如，Cedilla系統特別J編譯器即利用一型態式中介語言。但是，此編譯器係特定於Java來源語言，且因此並不需要處理多種或例如具有非型態安全之程式碼的語言。此外，此編譯器僅利用一組規則集合用以型態檢查，而因此並不適用於多個編譯層級。在研究社群裡，型態式中介語言通常會傾向高度地特定於來源語言，而不易對多種編譯階段進行工程處理(及設計各型態)。

【發明內容】

茲提供一種用以檢查在各式中介語言內之一致性的型態表現、型態檢查器、方法及編譯器。特定地說，該型態式中介語言適於運用在表現按多種(異質性)來源語言，這

包含型態式及非型態式語言、鬆散及強型態式語言，以及具與無垃圾回收功能的語言，所撰寫之程式。此外，可延伸該型態檢查器架構，以處置具有不同型態及原生運算的新式語言。型態表現、型態檢查器、方法及編譯器包含各種特點。可個別地及獨立地運用各項特點，或是可按各式組合及子組合的方式來運用各種特點。

在一特點裡，茲提供一種在編譯器內進行程式語言型態檢查作業之方法。一或更多規則集合會被採為對一型態檢查器的輸入，這會根據多項關鍵項之任一者，或二者以上的組合，來選擇一或更多規則集合。在這些之中，可為出現在被予以型態檢查之語言內的編譯作業、來源語言、架構及定態層級。接著會利用所選擇的一或更多規則集合對該語言進行型態檢查。

在另一特點中，一編譯器經供置有一型態檢查器，此者可根據多項關鍵項之任一者，或二者以上的組合，來建構一或更多規則集合。該等規則集合可包含一對應於強型態檢查之規則集合、一對應於弱型態檢查之規則集合、及一對應於表現型態檢查之規則集合。該弱規則集合可提供較高的定態作業彈性，像是提供型態編排，而該表現規則集合可拋除在中介程式表現各部分內的型態資訊。

在另一特點裡，提供一程式設計介面以建構複數個用以檢查一程式之中介表現的一致性之規則。該中介表現一致性的檢查作業可包含提供複數個規則給一型態檢查器，此者會根據預設關鍵項，將一第一組規則集合施用於一中

介表現，而將一第二組規則集合施用於另一中介表現。

按後載且參照於各隨附圖式之詳細說明，該等及其他特點即屬顯見。

【實施方式】

茲提供一種用以檢查在各式中介語言內之一致性的型態表現、型態檢查器及編譯器。該型態檢查器及該編譯器允許按照一程式元件之來源語言及/或編譯階段，運用不同的型態及型態檢查規則。例如，或希望將一高階最佳化器施用於按不同語言所撰寫的程式。這些語言或會具有不同的原生型態及原生運算。一語言可例如含有複數算術的型態及運算，而另一語言或含有特定於電腦圖學的型態及運算。藉由可讓不同系統將中介表現予以參數化，該最佳化器可適用於具不同原生型態及運算的語言。另一範例可為包含一程式，其中部分元件是按一語言之強型態式子集合所撰寫，而其他元件則是按完整語言所撰寫，而此非型態安全者。會希望是對該第一組的元件進行更多的錯誤檢查。這可藉由對於不同元件利用不同的型態檢查規則來完成。而又另一範例係在編譯過程中拋除型態資訊。該型態檢查器及編譯器允許在稍後階段裡拋除型態資訊，而同時強制在較早階段內保留著精確資訊。這可藉利用一未知型態併同於不同編譯階段的不同型態檢查規則來達成。

第 1 圖顯示一系統之一般編譯處理，該系統利用一具有不同較低層級以表現數種不同來源語言的型態式中介語

言。來源程式碼 100 - 106 係按四種不同來源語言所撰寫，這些可為或非屬型態式，且具有不同的型態強度層級。例如，按 C#所撰寫的來源程式碼 100 會為一遠強於例如按 C++所撰寫的來源程式碼 106 之型態式。來源程式碼會先予處理，然後由一讀取器 108 輸入該系統。然後，將該來源語言轉譯成一型態式中介語言的高階中介表現(HIR)。然後在區塊 110 處，依需要將該 HIR 加以分析及最佳化。接著，將該 HIR 轉譯成該型態式中介語言的一中階層中介表現(MIR)。此表現低於 HIR，但仍為與機器無關。在此時，可選擇性地將該 MIR 加以分析及最佳化，即如區塊 112 所示。然後在區塊 114 處藉由程式碼產生器，將該 MIR 轉譯成機器相關的型態式中介語言(LIR)低階表現。然後，在區塊 116 處，會選擇性地將 LIR 加以分析及最佳化，並在區塊 118 處將此供應給一發射器。該發射器會按照表示被讀入該系統內之來源程式碼的多種其一格式 120 - 126 來輸出程式碼。經此程序，就會將為完成該處理所必要的資料儲存在某種形式的持固性記憶體 128 內。

如此，編譯作業處理包含將各中介語言指令從一層級轉型到另一層級。例如，第 2 圖顯示將來源程式碼述句轉換到一 HIR 的作業，以及從 HIR 轉換到機器相關 LIR 的作業。可按多種高階語言來撰寫來源程式碼述句 200。這些語言係經設計，以便讓程式設計人員能夠依簡易瞭解的方式來撰寫及讀取程式碼。如此，程式設計人員能夠利用像是「+」的字元進行加法，並且能夠使用更有力的形式，

像是將兩個以上的運算元相加，即如述句 200 所示。

述句 202 - 206 為代表相同功能性之述句 200 的 HIR 表現，不過這是按更接近一電腦所能了解而又仍屬架構獨立之格式來表示。述句 202 利用一「ADD」指令來加一第一及第二變數，並將結果指配給一第一暫時變數 t1。接著，述句 204 利用另一「ADD」指令以將 t1 加到第三變數，然後將結果指配到一第一暫時變數 t2。然後，述句 206 利用一「ASSIGN」指令，將 t2 的數值指配到結果變數 z。

為實作型態檢查作業，該型態式中介語言會含有顯明地或內隱地表示的型態表現。一顯明型態表現會直接地在該表現內加以宣告。例如，該述句：

```
int a;
```

顯明地定義變數「a」為 int 型態。可藉由對某些程式碼述句定義一內定型態以內隱地表示一型態表現。例如，如對於一些函式之內定回返型態為 int，則該述句

```
f_start();
```

會宣告一函式 f_start()不取用參數，並且回返一 int 型態的數值。

一種對於在許多表現層級處，適用於多種程式語言之型態式中介語言的型態表現具體實施例可如附錄 A 所示。應了解此僅為各式可能具體實施例其一範例。

現回到附錄 A，這是在一型態類別階層裡定義許多型態表現，使得能夠藉由該型態式中介語言來表現各種語言的型態系統。對於所有的型態，會將一抽象基底類別定義

成為「Phx::Type」。該基底類別可例如在「sizekind」內含有對於像是真實、符號或未知(或可變)型態之各式型態的大小資訊。該基底類別亦可含有「typekind」，藉以設定型態分類作業。此外，可提供一外部型態作為一抽象型態，這可裹裝一外部定義型態，藉以提供從該型態式中介語言到原先之來源程式碼的後向對映。

在該基底類別下，一定義為「Phx::PtrType」之類別可表現指標型態。亦可定義各種指標。例如，一經管理且經垃圾收集之指標(指向一位於某個經垃圾收集之物件內的位置)、一經管理且未經垃圾收集之指標(指向一位於某個未經垃圾收集之物件內的位置)、一未經管理之指標(像是會在一例如按C++所撰寫之程式碼裡發現者)、一參考指標(指向一經垃圾收集之物件的基底)，及空無(null)。

在該階層架構的相同層級處，一經定義為「Phx::ContainerType」的類別可表示像是含有各內部成員之型態的容器型態。該等內部成員可具有欄位、方法及其他型態。一經定義為「Phx::FuncType」的類別可表示函式型態，包含任何必要的呼叫轉換。同時，一經定義為「Phx::Unmgd ArrayType」的類別可表示未經管理的陣列型態。在該階層架構內的「Phx:: ContainerType」底下，又可定義有四個類別。一經定義為「Phx::ClassType」的類別可表示類別型態，一經定義為「Phx::StructType」的類別可表示結構(struct)型態，一經定義為「Phx::InterfaceType」的類別可表示介面型態，而一經定義為「Phx::EnumType」的類別可

表示列舉型態。在該階層架構內的「Phx:: ClassType」底下，一另定義為「Phx::MgdArrayType」的類別可表示經管理之陣列型態。

在如附錄A的表現裡，一類別「primtype」被定義為一結構型態的特殊實例。「primtype」可包含各種型態，像是整數(int)、浮點(float)、未知、虛無(void)、條件程式碼、無號式int、xint等。在該型態式中介語言內的HIR及LIR兩者內都可運用這些表現。

此外，可將目標特定性的原生型態納入該型態表現內。有些語言具有複數算術型態，而如該型態系統了解之，則可效率地加以處理。即以「MMX」指令為例。這個指令係建置於某些版本的x86處理器內，而為支援在多媒體及通訊資料型態裡之單一指令/多重資料運算的額外指令集合其中一者。可自訂該型態系統，以藉最低的型態表現替換作業來辨識並運用這些指令。

附錄A內所顯示之各項型態的型態表現具體實施例亦可包含一「未知」型態，這可表示任意型態，並且依需要具有一相關於此者的大小。該大小係該數值之機器表現的大小。一未知型態可讓一編譯器藉由將該型態資訊從特定型態改變至一未知型態，而按控制方式拋除該型態資訊。這可讓編譯器根據所操縱之數值的大小來產生程式碼，即使是當該型態為未知時亦然。其他的型態可利用此未知型態，故未知型態也允許表現部分的型態資訊(其中為部分但並非所有的資訊為已知)。

例如，假定一指向一 int 型態的指標。在某些較低階段，會希望拋除型態資訊，int。未知型態可讓編譯器能夠替換該 int 型態為未知型態。然後型態檢查器會不需要檢查該所欲指標是否朝向一正確型態。基本上，這是利用在執行時間內按照不會負面地影響到該程式功能性之方式來處置所朝數值的機會。

另一種利用未知型態的範例則可如對於一函式定義一型態。若呼叫一具有一朝向未知型態指標之引數的函式，其中該引數先前具有朝於 int 的型態指標，該編譯器就必須相信所傳通者係一正確型態。將該指標予以解參指的結果或會或不會已知確為 int；然確會依如一 int 所運用。一更為複雜範例，可如在一虛擬函式呼叫從高階轉換到低階中介表現的過程裡引入一中介暫時性變數。於物件導向語言裡，廣泛地使用虛擬表(vtables)來實作虛擬呼叫。在低階中介表現裡製作虛擬呼叫的第一步驟係取獲一記憶體之物件的第一欄位。該第一欄位裡含有一個朝向一 vtable 的指標。然後將所取獲的結果指配給一暫時變數。該暫時變數之型態的建構作業(一表示一指向一 vtable 之指標的型態，其中該 vtable 可具有許多欄位)可為複雜且不易表示。反之，該編譯器可僅指配該中介暫時變數「朝向未知之指標」。如此，運用未知型態可簡化稍後的編譯階段，而其中並不需要保留詳細的型態資訊，否則對於該編譯器實作者或會表示一顯著負擔。

第 3 圖說明一編譯器系統具體實施例，此者可用以在

不同編譯階段裡對一型態式中介語言進行型態檢查，並且因此會在底下的不同階段裡對一型態式中介語言進行型態檢查。來源程式碼 300 代表各種來源語言之任一者。該來源程式碼 300 係被轉譯為一型態式中介語言的 HIR 302。藉此，該來源語言的型態表現會在該型態式中介語言之內被轉譯成該型態表現。

即按第 1 及 2 圖所解釋，該 HIR 在整個編譯處理裡會被降低。為加以說明，圖中繪出一高(HIR) 302、中(MIR) 304 及一低(LIR) 306 層級表現。然而，該具體實施例並非受限於此。可對任意數量的編譯階段進行型態檢查。

可由該型態檢查器 308 在各表現層級處對該中介語言進行型態檢查。該型態檢查器 308 實作一演算法或程序，以將一或更多規則集合 310 施用於該編譯處理各階段，並因此可施用於該中介語言的各表現。該規則集合 310 為經設計以改變像是來源語言、編譯階段、型態強度等等之語言性質的規則集合。

例如，假定來源程式碼 300 含有按 C++程式語言所撰寫的程式碼。該 C++來源程式碼 300 會首先被轉譯成為該型態式中介語言 HIR 302。如有必要，此時該型態檢查器 308 可與該 HIR 302 互動，藉以決定任意數量的性質。這些性質可包含編譯階段(HIR)、此來源程式碼型態(C++)、該語言是否為型態式(是的)、為鬆散或強型態式(鬆散)等。根據這些性質，該型態檢查器可選擇一組適當的規則集合。一旦選擇一規則集合後，該型態檢查器會按照該組

規則集合來對該 HIR 進行型態檢查。一旦將該 HIR 降低至 MIR 或 LIR 之後，就會再次接取該等性質，而可能是同一或不同組的規則集合為適當者。

在一具體實施例裡，可將三組型態檢查規則提供給該型態檢查器。一組可對應於「強」型態檢查作業，像是型態檢查 C#或 MSIL 時所希望者。另一組可對應於「弱」型態檢查作業，這會是比起強型態檢查為鬆散的型態檢查。例如，弱型態檢查規則集合可提供型態編排。一型態編排是當一型態的變數被令依某單一運用而如另一型態進行動作時。例如，一 int 型態的變數可被令為如 char (字元)而動作。下列程式碼利用一型態編排以列印字母「P」:

```
int a;
a=80;
cout<<(char) a;
```

如此，即使是「a」被定義成 int 型態，且經指配數值 80，該 cout 述句會因型態編排之故而將該變數「a」視為 char 型態，並因此顯示一「P」(ASCII 數值 80)而非 80。

最後，一組可對應於「表現」檢查。該「表現」檢查允許拋除該中介程式表現之各部分內的型態資訊，像是藉利用一未知型態，且可包含指示此種型態資訊何時可予拋除或是一未知型態可何時替代另一型態之規則。例如，可禁止將回返 Void 型態數值之函式的結果指配給一未知型

態的變數。

此外，可在編譯作業的單一階段裡運用一組以上的規則集合。例如，假定該來源程式碼300含有一單一語言，但是含有強型態式的區段及鬆散型態式的區段。該型態檢查器可在某些強型態式區段對於HIR利用一組規則集合，而在一些鬆散型態式之程式碼區段利用另一組規則集合。

第4圖係一運用在一類似如第3圖所繪之編譯器系統的型態檢查器區塊圖。該型態檢查器400可接受對應於不同來源語言及/或不同編譯階段之任意數量的規則集合以做為輸入。在第4圖裡，對該型態檢查器400提供有四個規則集合402-408。規則集合402代表對一具有強定態之語言的HIR之規則集合，規則集合404代表對一具有弱定態之語言的HIR之規則集合，規則集合406代表對一不具定態的HIR之規則集合，而規則集合408代表對一LIR的規則集合。程式模組410代表一在HIR裡具強定態之語言，而程式模組412代表在被降至一LIR後的程式模組410。

該型態檢查器400會根據經型態檢查之程式模組的性質選擇一適當規則集合，並利用一併入程序或演算法將所選擇的規則集合施用於該程式模組。例如，該型態檢查器400可選擇該規則集合402(代表對一具有強定態之語言的HIR之規則集合)藉以型態檢查程式模組410(代表一在HIR裡具強定態之語言)。然後，該型態檢查器400可選擇該規則集合408(代表對一LIR之規則集合)以型態檢查該

程式模組 412（代表一在 LIR 裡具強定態之語言）。

第 5 圖為一流程圖，此係用以選擇一待由該型態檢查器所施用之規則集合的可能程序具體實施例。在區塊 500，一型態檢查器在一區段內讀取來源程式碼的型態式中介表現，並必須選擇一規則集合以進行型態檢查作業。決策 502 決定該型態式中介語言究係一 HIR、MIR 及 LIR。

如此係一 HIR 或一 MIR，則會處理該決策 504 以決定原始的來源程式碼究係鬆散或強型態式。若此為一鬆散型態式，則會處理該區塊 506 以選擇一對應於弱型態檢查作業之規則集合。如該者為強型態式，則會處理該區塊 508 以選擇對應於強型態檢查作業的規則集合。

倘此係一 LIR，則會處理該決策區塊 510 以選擇一對應於表現態檢查作業的規則集合。應注意到該第 5 圖僅為一具體實施例。確可對應於並基於不同性質而選擇任意數量的規則集合。

所描述之型態檢查系統的規則集合可簡易地延伸到整個新式語言，且亦擴至現有語言的新式特性。例如，若須引入一新語言，則僅須對該新語言撰著一新規則集合。由於規則集合區別於該型態檢查器或編譯器系統本身，且係經設計以按如各個整體而接受各規則集合，因此可配發新式語言的規則集合，而無須重新配佈或更新現有型態檢查作業系統或編譯器。同樣地，假使將一新式特性增入於一現有語言，例如像是對 C++增加 XML 支援，則可簡易地按動態方式重新組態設定在各編譯階段處對應到 C++的規則

集合，以利處置該新式特性。再次地，無須更新或重新配發新的核心系統。

各規則集合也可提供各型態的限制項。例如，當一類別從另一者而繼承時，是否能夠對一特定型態提供子定態處理，可為各規則內所描述的限制項。另一限制項可為盒框限制項，像是或希望指示資料可被轉換成一含有該資料的虛擬表格。其他可包含一大小限制項，或一表示相同原生型態之必要性的原生型態限制項。就像該規則集合的任何其他部分，可視需要增加新的限制項。

可透過一對一用以撰著該等規則集合之應用程式的程式設計介面，來建構型態檢查器所運用的規則集合。該應用程式可建構各規則，使得該規則集合按原生型態之階層的方式表現，且各規則會被指配給該型態式中介語言的個別指令。可按型態圖的形式提供該階層，此者可顯明地表示與一特定程式模組或編譯單元相關的各種型態元件。可將像是符號及運算的各 IR 元件關聯於各型態系統的各元件。該等型態圖節點可描述原生及經建構之型態，以及該等關係，像是元件，巢狀型態、函式簽章、介面型態、階層元件及其他資訊，像是來源名稱以及指向外部型態元件之模組/組裝的參考。

一簡易型態規則的範例如下：

ADD

N=add n, n

假設對此範例，I 係一有號整數型態，U 為一無號整

數型態，X為任一整數型態，F為浮點且N為上述任者。第6圖顯示這些型態之間的階層關係。型態N位於該階層的頂端。型態F及X從該型態N向下分支，構成該階層的後續層級。最後，型態U及I從X型態向下分支，構成該階層的最低層級。如此，對一「ADD」中介語言指令，根據此規則，僅N或該階層的較低者可由該add指令進行處理，且運算元在該階層裡不可高於該結果。例如，兩個整數可相加而產生一整數(I=ADD i, i)，或一整數及一浮點可相加以而產生一浮點(F=ADD i, f)。然而，一浮點及一整數無法相加而產生一整數(I=ADD i, f)。

以階層方式表現原生型態可讓規則集合能夠簡易地更換。在過去，型態規則常常是利用來源程式碼而按程式設計方式所表現。例如，一型態檢查器可含有大量實作各項型態檢查器規則的交換(switch)述句。因此，改變一規則會需要修改該型態檢查器的來源程式碼。不過，階層式規則集合提供較簡易的延伸性。考慮先前對於ADD指令的規則。如一發展人員有意對一複數型態增加某一型態，例如C，可在如第7圖所示之階層內的N型態底下逕行增加，且對ADD指令的規則並不需要替換以依需要運作。

一種按一型態規則，在一型態檢查系統中檢查一指令的方法可如圖8所示。首先，處理區塊800以按文意方式檢查該指令。如此，在806考慮該指令，該型態檢查器會根據ADD指令的型態規則，確保正確數量的來源及目的地表示存在(例如在本例中，2個來源表示及一個目的地表

示)。在該中介表現裡，各個表示(以及子表示)可於其上具有一顯明型態。然後在區塊 802 處理，該型態檢查器會真實地辨識 e1、e2 及 foo (e3)的顯明型態相符於 ADD 指令的型態規則。在區塊 804，如有必要，該型態檢查器會行旅子層級以進一步對各指令進行型態檢查。例如，該型態檢查器可檢查該等表示 e1、e2 及 foo (e3)與該等的顯明型態相一致。例如，該型態檢查器可檢查該 foo 具有一函式型態。這可檢查該函式型態的結果型態會與 foo 上的顯明型態相同。這可進一步檢查有單一引數型態，且該型態 e3 相符於該型態。這可確保呼叫到 e3 的型態會與型態規則一致。

第 9 圖說明一電腦系統範例，該者可作為一型態檢查系統之具體實施例的作業環境。該電腦系統包含一個人電腦 920，包括一處理單元 921、一系統記憶體 922 及一系統匯流排 923，此者可將包含該系統記憶體在內的各種系統元件互連至該處理單元 921。該系統匯流排可包含多種型態結構的任者，此結構包含一記憶體匯流排或記憶體控制器、一週邊匯流排，以及一利用一匯流排架構，即如像是 PCI、VESA、「微通道(MCA)」、ISA 及 EISA 之區域匯流排。該系統記憶體包含唯讀記憶體(ROM) 924 及隨機存取記憶體(RAM) 925。一含有基本副程式，而可協助像是在開機過程裡於該個人電腦 920 內之各元件間傳送資訊的基本輸入/輸出系統會被存放在內 ROM 924 內。該個人電腦 920 進一步包含一硬碟機 927、一即如對一可移除碟片 929 進

行讀取或寫入之磁碟機 928，以及一即如可讀取 CD-ROM 碟片 931 或者是用以讀取或寫入至其他光學媒體之光碟機 930。該硬碟機 927、磁碟機 928 及光碟機 930 分別地藉一硬碟機介面 932、磁碟機介面 933 及一光碟機介面 934 而連接於該系統匯流排 923。這些碟機與該等相關電腦可讀取媒體提供資料、資料結構、電腦可執行指令(像是動態鏈結庫及可執行檔案的程式碼)等等的非揮發性儲存，例如個人電腦 920。以上關於電腦可讀取媒體的描述雖參照於硬碟、一可移除磁碟片及一 CD，然這也可包含其他種類而可由一電腦讀取的媒體，像是磁匣、快閃記憶卡、數位視訊碟、白努力(Bernoulli)卡匣等等。

可將多個程式模組儲存在各碟機及 RAM 925 內，包含一作業系統 935、一或更多應用程式 936、其他程式模組 937 以及程式資料 938。一使用者可透過一鍵盤 940 和像是滑鼠 942 之點指裝置，將命令與資訊輸入到該個人電腦 920 內。其他輸入裝置(未以圖示)可包含一麥克風、搖桿、遊戲板、衛星碟、掃描器等。這些及其他輸入裝置通常是會透過耦接於該系統匯流排的序列埠介面 949，然亦可藉由其他如平行埠、遊戲埠或一通用序列匯流排(USB)之介面所連，而連接到該處理單元 921。一監視器 947 或其他種類的顯示裝置也可，透過一像是顯示控制器或視訊配接器 948 之介面而連接到該系統匯流排 923。除監視器外，個人電腦通常包含像是喇叭及印表機的其他週邊輸出裝置(未以圖示)。

該個人電腦920可利用至一或更多遠端電腦，即如遠端電腦949，之邏輯連接而運作於一網接環境裡。該遠端電腦949可為一伺服器、一路由器、一端點裝置或其他常見網路節點，並且通常會包含許多或所有前文按照個人電腦920所描述的元件，然第9圖中僅描繪出一記憶體儲存裝置950。第9圖所描繪的各邏輯連接包含一區域網路(LAN)951，及一廣域網路(WAN)952。此等網接環境常見於辦公室、企業界電腦網路、企業內網路及網際網路。

當用於一LAN網接環境下時，該個人電腦920會透過一網路介面或配接器953而連接到區域網路951。當常用於一WAN網接環境下時，該個人電腦920通常會包含一數據機或其他裝置以於該廣域網路952，像是網際網路，上建立通訊。該數據機954，可為內插式或外接式，會透過序列埠介面946而連接到該系統匯流排923。在一網接環境裡，相對於該個人電腦920或其局部所描述的程式模組，可儲存在該遠端電腦儲存裝置內。所示之網路連接僅屬範例，且可運用其他在各電腦間建立通訊鏈路的方式。

既已說明及描述所述具體實施例之原理，熟諳本項技藝之人士應即瞭解確可修改各具體實施例其排置方式與細節，而無虞悖離本發明原理。

例如，一本揭具體實施例描述可供應給一型態檢查器或一編譯器的一或更多規則集合，使得該編譯器或型態檢查器選擇一或更多規則集合，以根據所予型態檢查之語言及/或編譯階段對該語言進行型態檢查。然而，在替代方式

裡，可提供一單一組規則集合給一型態檢查器或編譯器，使得該編譯器或型態檢查器能夠根據所予型態檢查之語言及/或編譯階段，靜態地或於執行時間的動態方式，從該單一規則集合建構出一或更多的規則子集合。

鑒於許多可能具體實施例，應了解所述具體實施例僅包含範例，而不應被視為本發明範圍之限制。相反地，本發明係由後載申請專利範圍所定義。從而吾人主張歸屬該等申請專利範圍內之所有具體實施例皆屬本發明。

【圖式簡單說明】

第1圖係一一般編譯處理之流程圖。

第2圖係一表格，此者顯示一來源程式碼述句至一高階表現然後再至一機器相關之低階表現的轉換方式。

第3圖係一資料圖式，此圖說明一用以在編譯作業的各種階段處，進行一型態式中介語言型態檢查之編譯器系統的具體實施例。

第4圖係一用於一編譯器系統之型態檢查器的區塊圖。

第5圖係一流程圖，此係用於一選擇待由一型態檢查器所施用之規則集合的可能程序。

第6圖係一顯示於各型態間之階層式關係的直接繪圖。

第7圖係一顯示將一型態增至各型態間之階層式關係的直接繪圖。

第 8 圖係一流程圖，此為用以在一型態檢查系統內，針對一規則集合而檢查一指令之方法。

第 9 圖係一電腦系統範例區塊圖，此者可作為一用於一型態檢查系統具體實施例之作業系統。

【元件代表符號簡單說明】

100 來源程式碼　　102 來源程式碼

104 來源程式碼　　106 來源程式碼

108 讀取器　　110 高階分析及最佳化

112 機器無關最佳化　　114 程式碼產生作業

116 機器相關最佳化　　118 發出

120 MSIL 格式　　122 .OBJ 格式

124 .EXE 格式　　126 .PDB 格式

128 持固性資料結構　　300 來源程式碼

302 HIR　　304 MIR

306 LIR　　308 型態檢查器

310 規則集合　　400 型態檢查器

402 具有強定態之語言的 HIR 之規則集合

404 具有弱定態之語言的 HIR 之規則集合

406 不具定態的 HIR 之規則集合

408 對 LIR 的規則集合　　410 具強定態之語言 HIR

412 具強定態之語言 LIR

500 在一區段內讀取來源程式碼的型態式中介表現

502 該型態式中介語言為按 HIR、MIR 或 LIR？

504 原始來源程式碼為強或弱型態式？

506 (弱)選擇一對應於弱型態檢查之規則集合

508 (強)選擇一對應於強型態檢查之規則集合

510 選擇一對應於表現型態檢查之規則集合

800 按照各型態規則以語意方式檢查各指令

802 按照各型態規則來檢查對於各表示的顯明型態

804 如有必要，行旅到各次層級以進一步型態檢查指令

920 個人電腦　　921 處理單元

922 系統記憶體　　923 系統匯流排

924 唯讀記憶體(ROM)　　925 隨機存取記憶體(RAM)

925 基本輸出/輸入副程式(BIOS)

927 硬碟機　　928 磁碟機

929 可移除碟片　　930 光碟機

931 CD-ROM 碟片　　932 介面

933 介面　　934 介面

935 作業系統　　936 應用程式

937 程式模組　　938 程式資料

940 鍵盤　　942 滑鼠

946 序列埠介面　　948 視訊配接器

950 記憶體儲存裝置　　951 區域網路(LAN)

952 廣域網路(WAN)　　953 配接器

954 數據機

十、申請專利範圍：

1. 一種根據一或更多規則集合在一編譯器中對按一來源語言編寫之一來源碼進行型態檢查的方法，該方法包含以下步驟：

基於該來源碼轉譯為中介表現之一編譯階段，選擇一或更多的規則集合用於對該來源碼進行型態檢查；以及

基於該選擇之一或更多規則集合，對該來源碼進行型態檢查；

其中該一或更多的規則集合包含一對應於強型態檢查之規則集合、一對應於弱型態檢查之規則集合，及一對應於表現型態檢查之規則集合。

2. 如申請專利範圍第 1 項所述之方法，其中對該來源碼進行型態檢查包含對該來源碼之複數中介表現的各者進行型態檢查。

3. 如申請專利範圍第 2 項所述之方法，其中所選擇的一或更多規則集合係因各個表現而異。

4. 如申請專利範圍第 1 項所述之方法，其中該等中介表現之之至少一者包含一型態，其指示該來源碼之一元件可

為複數型態之一者。

5. 如申請專利範圍第 4 項所述之方法，其中該等一或更多的規則集合包含用以對一型態進行型態檢查之規則，而該型態指示該來源碼之一元件可為複數型態中之一者。

6. 如申請專利範圍第 1 項所述之方法，其中該等一或更多的規則集合包含複數具有一階層化格式之規則。

7. 如申請專利範圍第 1 項所述之方法，其中該表現型態檢查允許用於該來源碼之元件的拋除型態資訊。

8. 如申請專利範圍第 1 項所述之方法，其中該弱型態檢查允許型態編排。

9. 如申請專利範圍第 1 項所述之方法，其中選擇一或更多的規則集合亦基於該來源語言，該方法更包含以下步驟：

決定該來源碼之編譯階段，其中該編譯階段係代表一高層級中介表現之一第一階段、一中層級中介表現之一第二階段及一低層級中介表現之一第三階段中之一者；

當該編譯階段是該第一階段或該第二階段，若該來

源碼之來源語言是強型態，選擇對應於強型態檢查之規則集合，若該來源碼之來源語言是弱型態，選擇對應於弱型態檢查之規則集合；及

當該編譯階段是該第三階段，選擇對應於表現型態檢查之規則集合。

10.一種含有電腦可執行指令之電腦可讀取媒體，以實作如申請專利範圍第 1 項所述之方法。

11. 一種於一編譯器中對按一型態式中介表現所表現之來源碼進行型態檢查的方法，該方法包含以下步驟：

從不同規則集合中，基於該型態式中介表現而選擇一規則集合，其中選擇一規則集合係基於一編譯程序之一系列步驟中產生該型態式中介表現之步驟；以及

基於該選擇規則集合，對該型態式中介表現進行型態檢查；

其中該等不同規則集合包含一對應於強型態檢查之規則集合、一對應於弱型態檢查之規則集合，及一對應於表現型態檢查之規則集合。

12. 如申請專利範圍第 11 項所述之方法，其中選擇一規則

集合係基於一產生該型態式中介表現的來源語言。

13. 如申請專利範圍第 11 項所述之方法，其中選擇一規則集合係基於與該型態式中介表現之一或更多特定指令相關之一處理器架構。

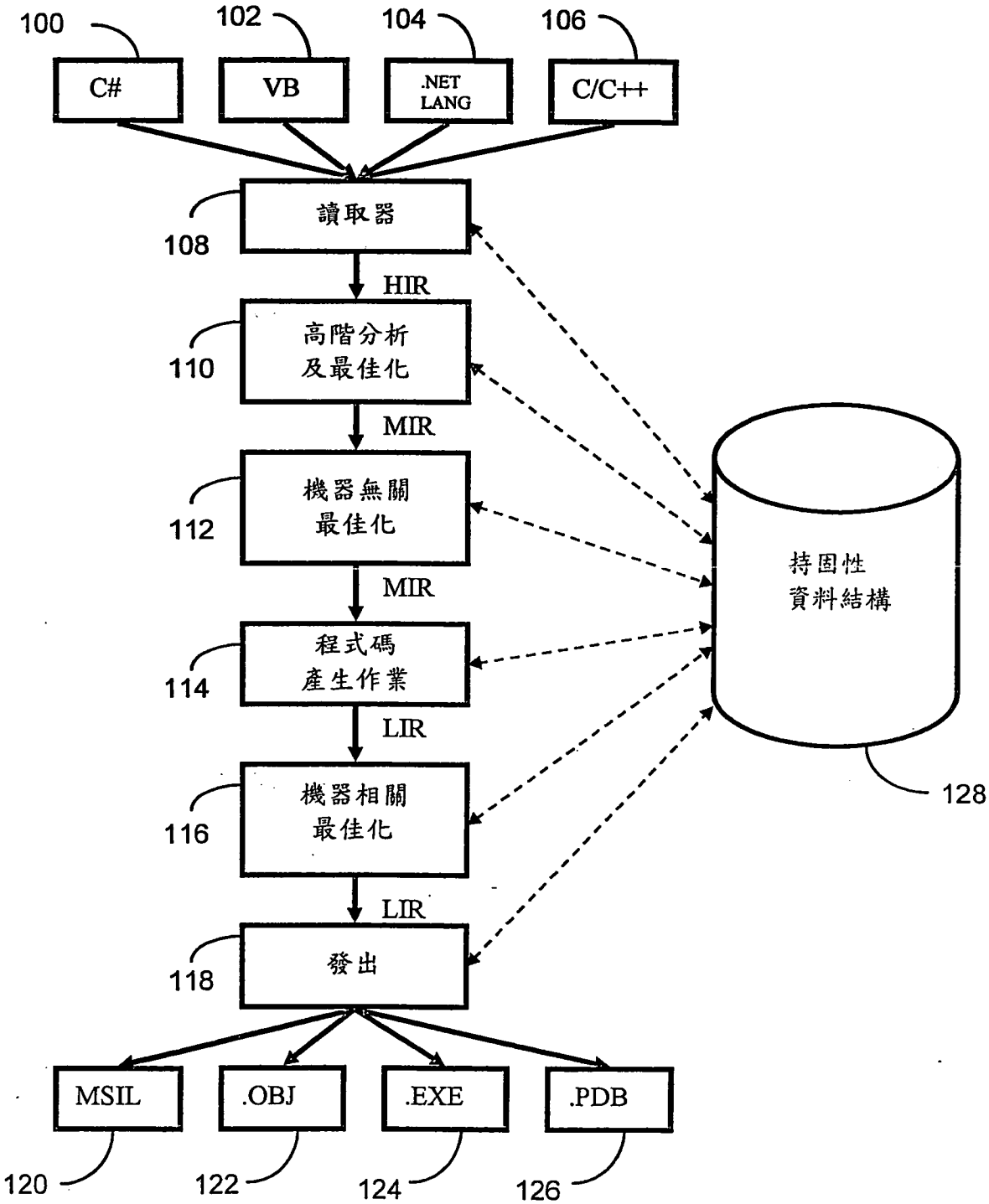
14. 如申請專利範圍第 11 項所述之方法，其中選擇一規則集合係基於該型態式中介表現是否代表經辨識或未經辨識碼。

15. 如申請專利範圍第 11 項所述之方法，其中選擇一規則集合係基於該型態式中介表現是否代表型態安全碼，或是非型態安全碼。

16. 如申請專利範圍第 11 項所述之方法，其中選擇一規則集合係基於該型態式中介表現是否代表型態式或是非型態式碼。

17. 一種含有電腦可執行指令的電腦可讀取媒體，以實作如申請專利範圍第 11 項所述之方法。

第 1 圖

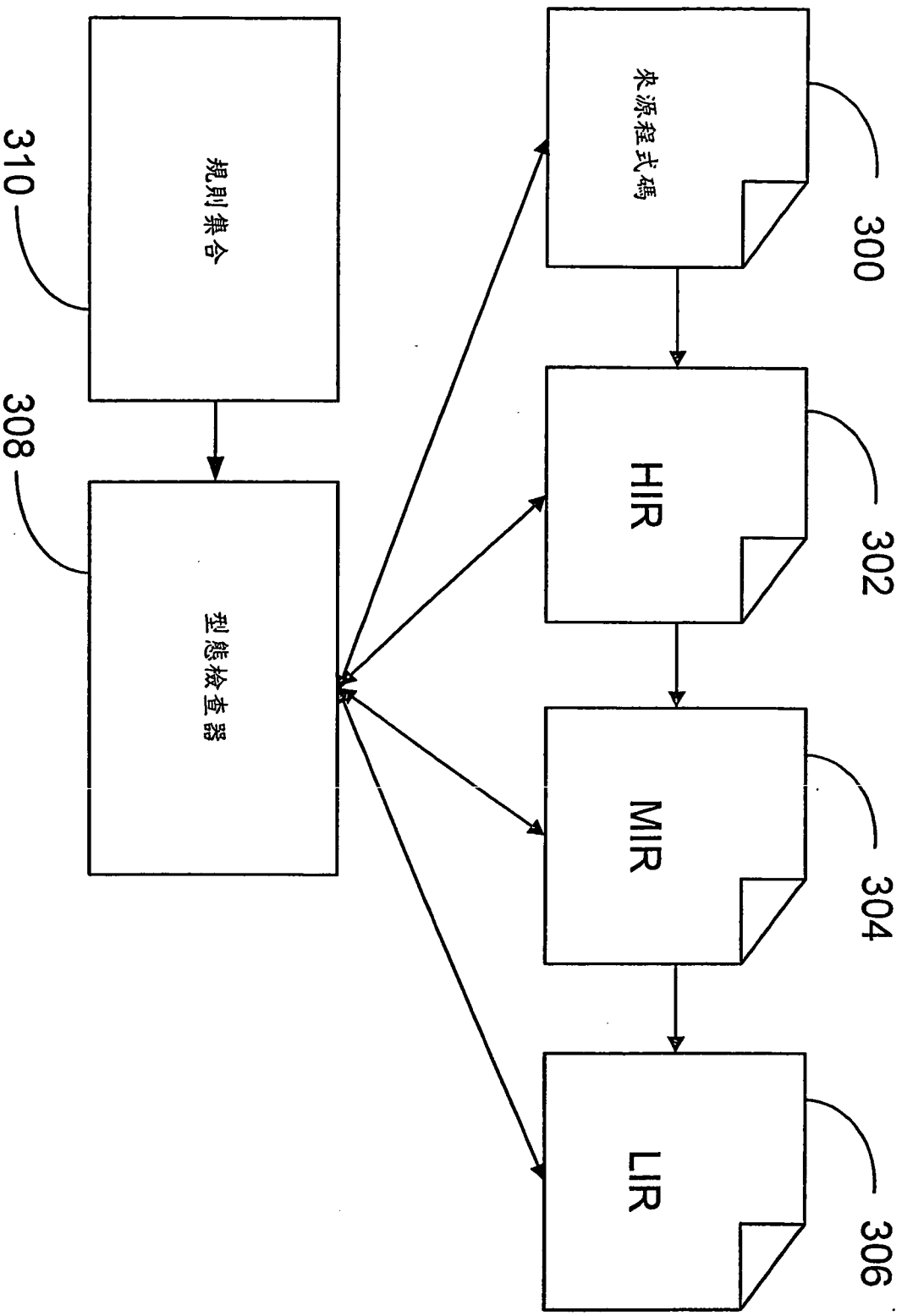

100
102
104
106
C#
VB
.NET LANG
C/C++
讀取器
108
HIR
高階分析及最佳化
110
MIR
機器無關最佳化
112
MIR
程式碼產生作業
114
LIR
機器相關最佳化
116
LIR
發出
118
持固性資料結構
128
MSIL
.OBJ
.EXE
.PDB
120
122
124
126

第 2 圖

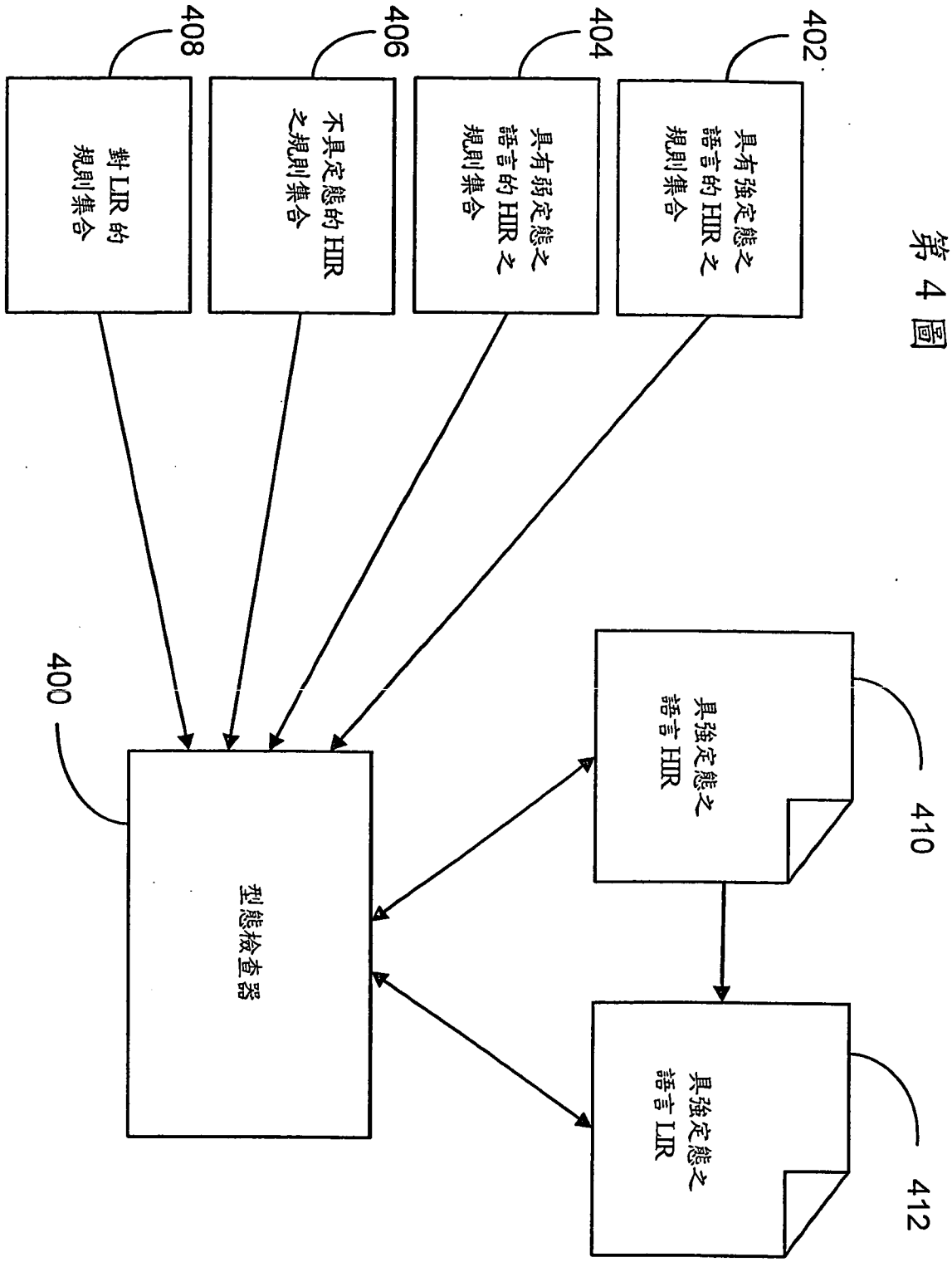
來源：

```
                z = a + b + c;   - 200
HIR:
                t1 = ADD    a, b   - 202
                t2 = ADD    t1, c  - 204
                 z = ASSIGN t2     - 206
LIR:
      t1(EAX), cc = x86add a(EAX), b(EDX)  - 208
      t2(EAX), cc = x86add t1(EAX), c(EBX) - 210
                 z = x86mov t2(eax)        - 212
```

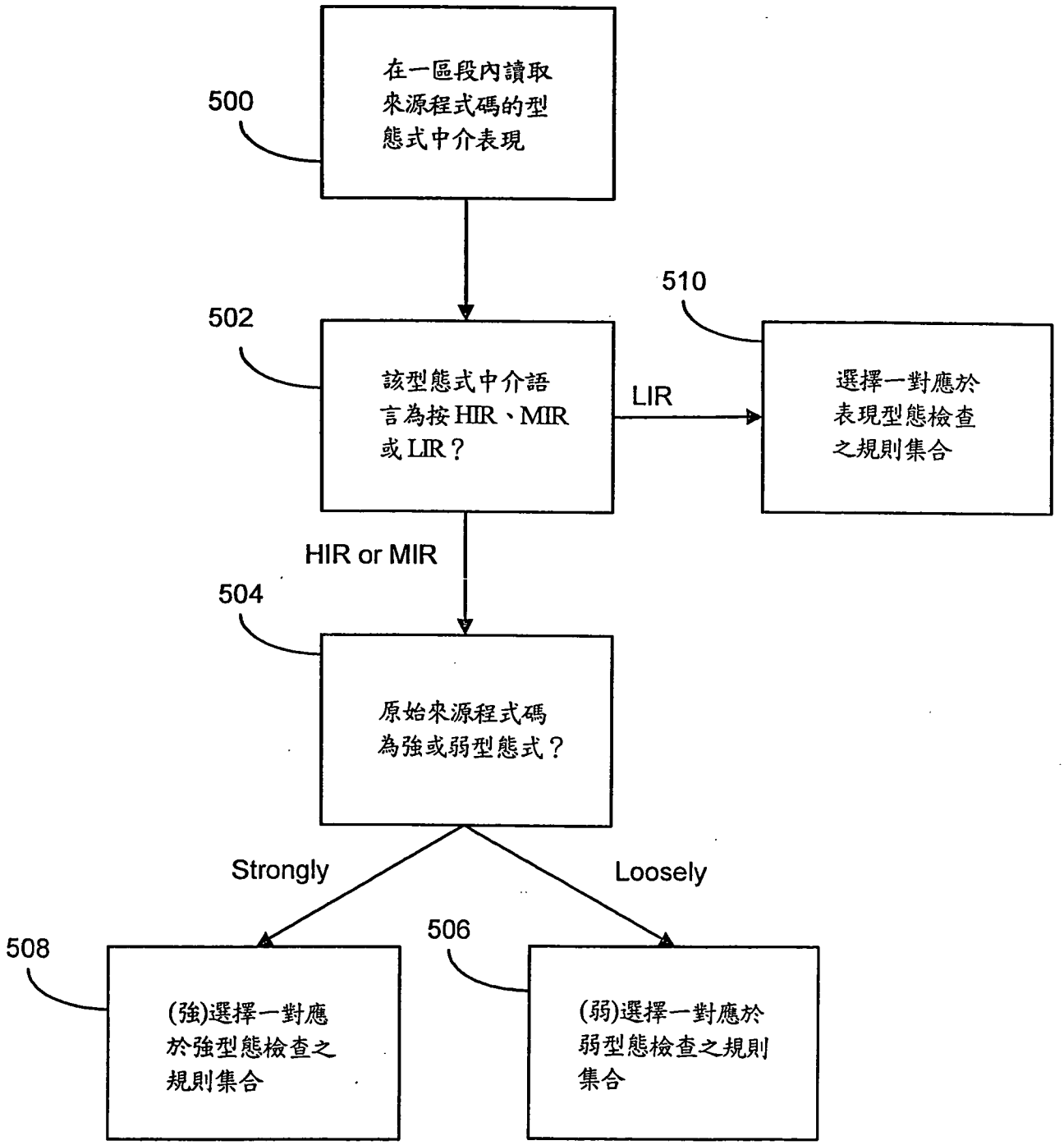
第 3 圖



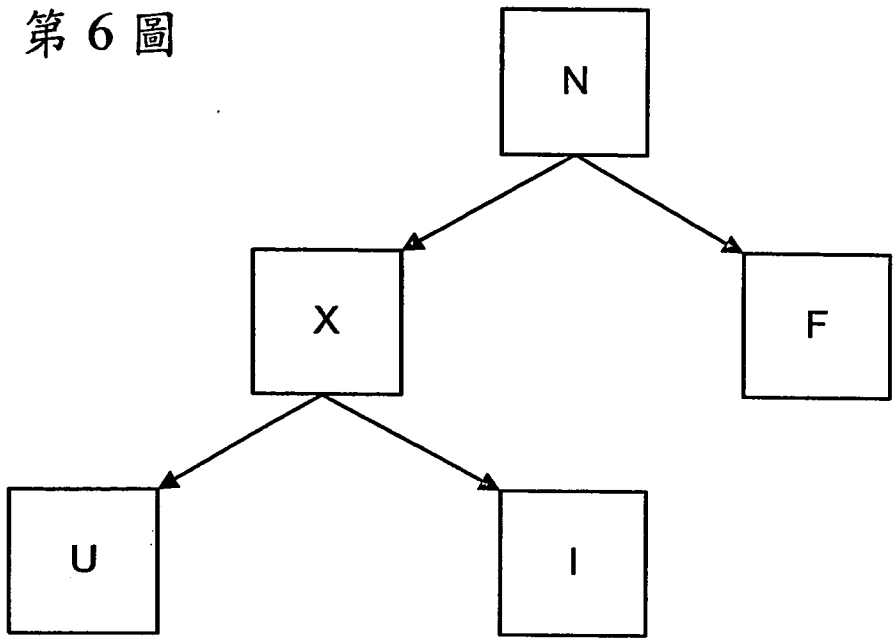
第 4 圖


402
具有強定態之語言的 HIR 之規則集合
404
具有弱定態之語言的 HIR 之規則集合
406
不具定態的 HIR 之規則集合
408
對 LIR 的規則集合
400
型態檢查器
410
具強定態之語言 HIR
412
具強定態之語言 LIR

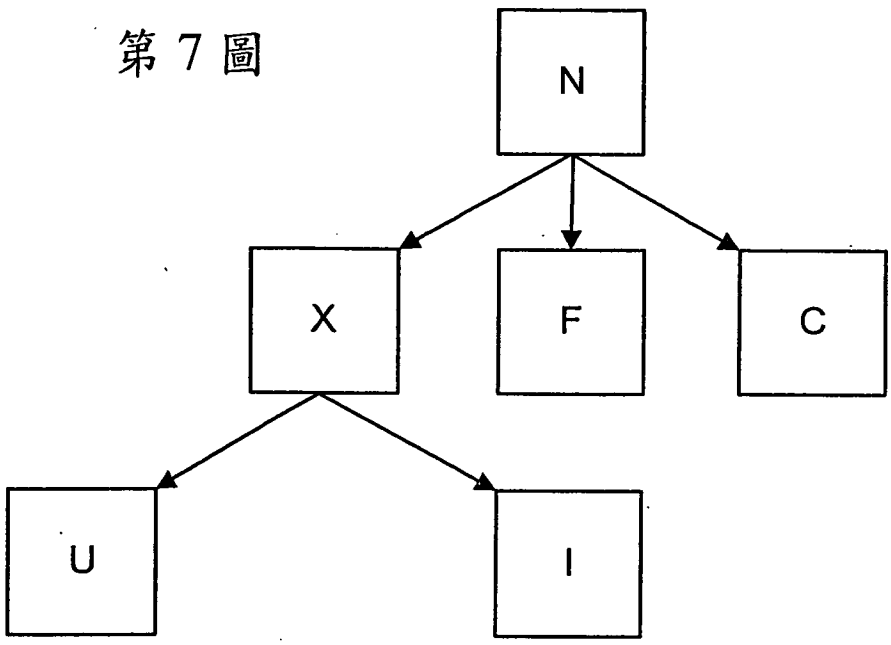
第 5 圖


500
在一區段內讀取來源程式碼的型態式中介表現
502
該型態式中介語言為按HIR、MIR或LIR？
LIR
510
選擇一對應於表現型態檢查之規則集合
HIR or MIR
504
原始來源程式碼為強或弱型態式？
Strongly
Loosely
508
(強)選擇一對應於強型態檢查之規則集合
506
(弱)選擇一對應於弱型態檢查之規則集合

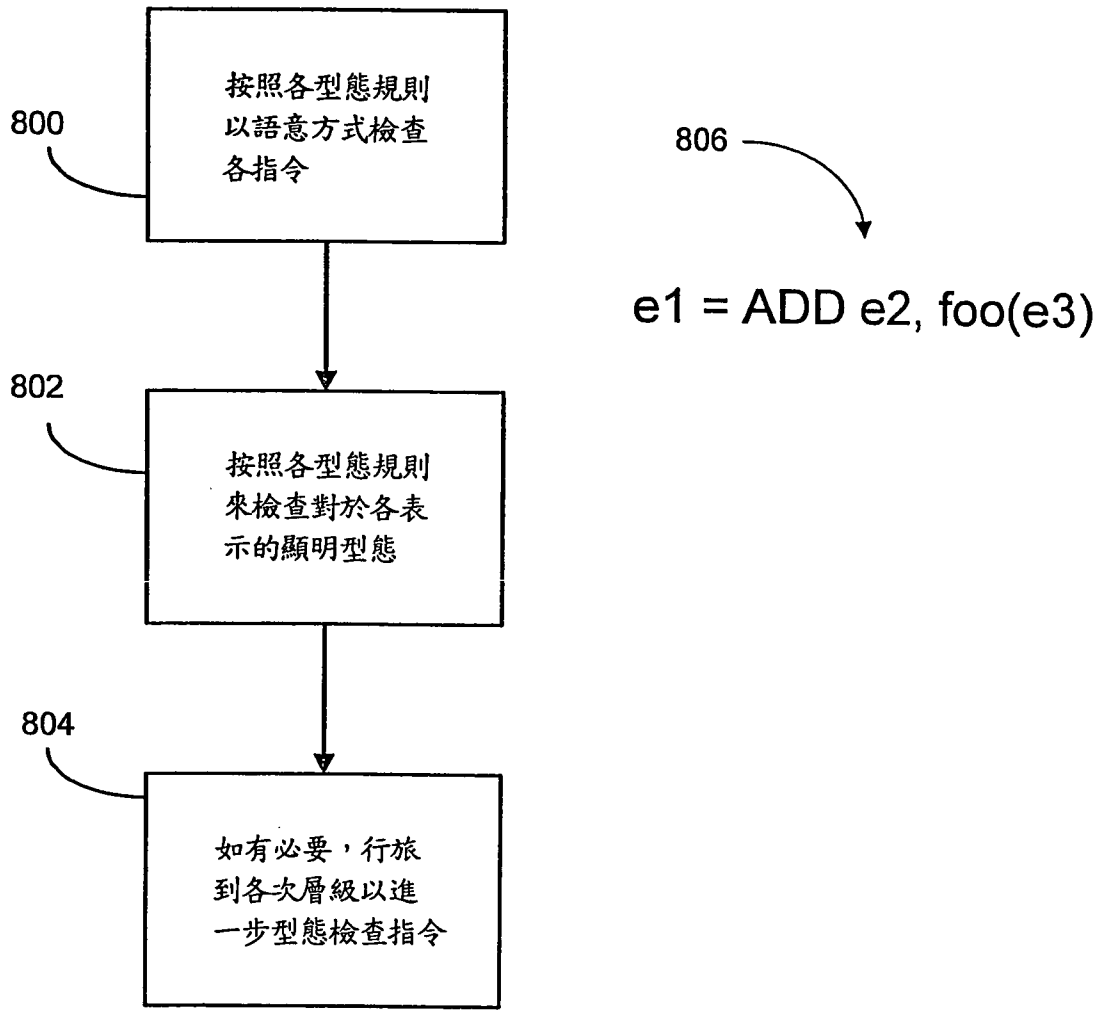
第 6 圖



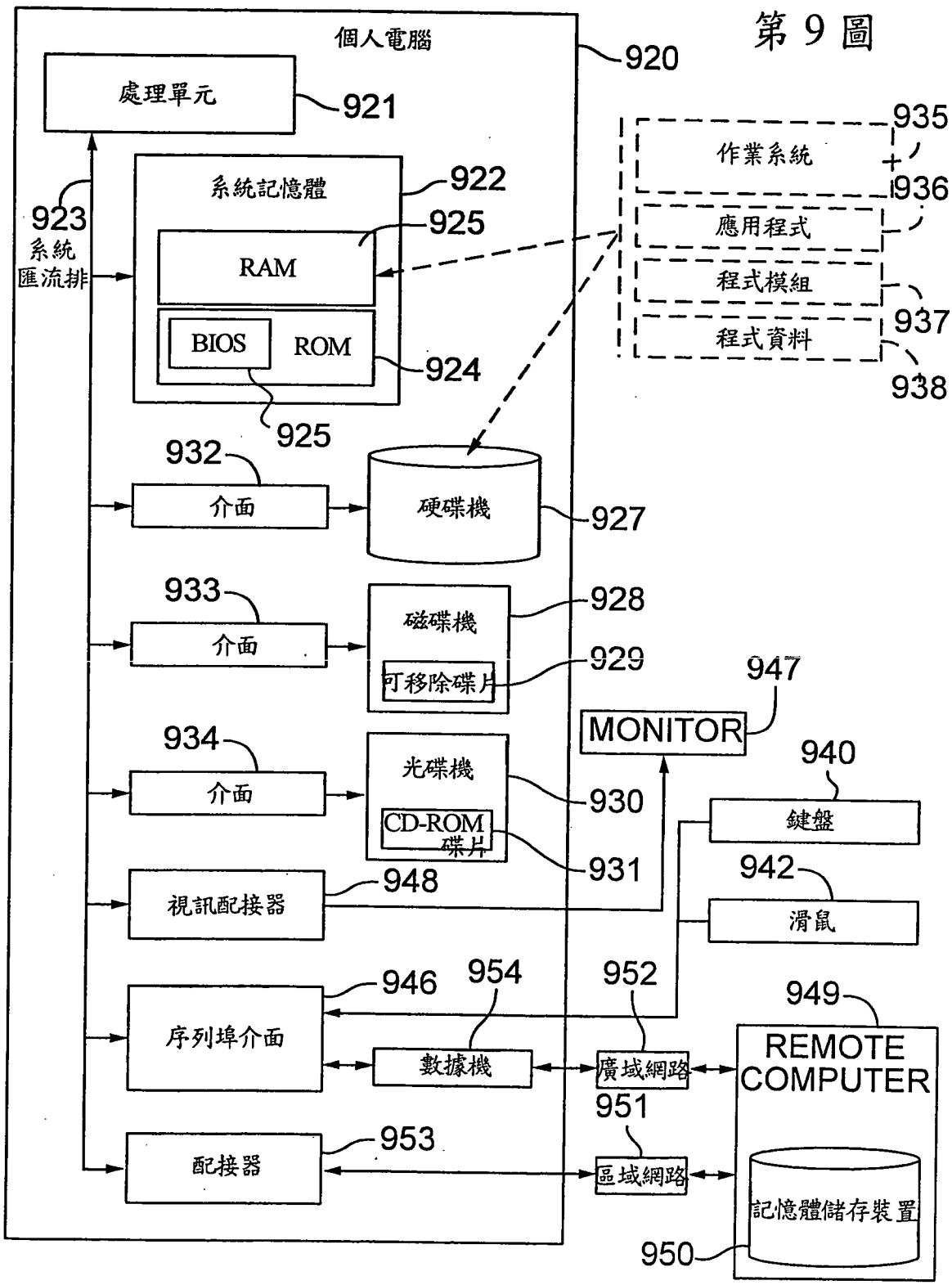
第 7 圖



第 8 圖


800
按照各型態規則以語意方式檢查各指令
802
按照各型態規則來檢查對於各表示的顯明型態
804
如有必要，行旅到各次層級以進一步型態檢查指令
806
e1 = ADD e2, foo(e3)

第 9 圖


個人電腦
920
處理單元
921
系統記憶體
922
923
系統
匯流排
RAM
925
BIOS
ROM
924
925
作業系統
935
936
應用程式
程式模組
937
程式資料
938
932
介面
硬碟機
927
933
介面
磁碟機
928
可移除碟片
929
934
介面
光碟機
930
CD-ROM
碟片
931
947
MONITOR
940
鍵盤
942
滑鼠
948
視訊配接器
946
954
952
949
序列埠介面
數據機
廣域網路
REMOTE
COMPUTER
951
953
配接器
區域網路
記憶體儲存裝置
950