[19]中华人民共和国国家知识产权局

[51]Int. Cl[6]
G06F 9/46

[12]发明专利申请公开说明书

[21]申请号 98122521.7

[43]公开日 1999年6月2日

[11]公开号 CN 1218223A

[22]申请日 98.11.19 [21]申请号 98122521.7
[30]优先权
[32]97.11.26 [33]JP [31]324945/97
[71]申请人 国际商业机器公司
地址 美国纽约
[72]发明人 铃木俊宏 南和宏

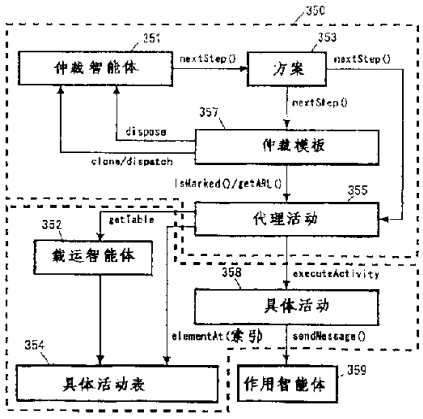
[74]专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 酆 迅

权利要求书3页 说明书32页 附图页数33页

[54]发明名称 执行移动对象的方法以及存储移动对象的记录介质

[57]摘要

为移动到各地点上把定义着要在各地点上执行的作业的具体活动358放在载运智能体352上载运。另一方面，移动对象的实体350持有代理活动355，后者向载运智能体保持的具体活动发送执行指令。移动对象的实体350发送指令，以便执行移动到各地点的具体活动358。



ISSN1008-4274

权　利　要　求　书

1．一种执行移动对象的方法，以在网络上分布的不同地点处执行不同的作业，该方法特征在于包括步骤:

(a) 生成第一载运智能体，其持有定义着要在第一地点执行的一个作业的第一具体活动;

(b) 生成第二载运智能体，其持有定义着要在第二地点执行的一个作业的第二具体活动;

(c) 生成移动对象实体，其含有说明所述第一载运智能体的信息并含有说明所述第二载运智能体的第二信息，和指示所述第一载运智能体在所述第一地点执行所述第一具体活动，并指示所述第二载运智能体在所述第二地点执行所述第二具体活动;

(d) 把所述第一载运智能体移动到所述第一地点;

(e) 把所述第二载运智能体移动到所述第二地点;

(f) 把所述移动对象的实体移动到所述第一地点;

(g) 通过使所述移动对象的实体说明所述第一载运智能体和指示所述第一具体活动执行作业，在所述第一地点执行所述第一具体活动的作业;

(h) 把所述移动对象的实体移动到所述第二地点；以及

(i) 通过使所述移动对象的实体说明所述第二载运智能体和指示所述第二具体活动执行作业，在所述第二地点执行所述第二具体活动的作业。

2．一种执行移动对象的方法，用于在网络上分布的至少一个地点上执行一个作业，该方法特征在于包括步骤:

(a) 生成一个载运智能体，其持有定义着要在所述地点上执行的一项作业的一个具体活动;

(b) 生成移动对象的实体，其含有说明所述载运智能体的信息，并指示所述载运智能体在所述地点执行所述具体活动;

(c) 把所述载运智能体移动到所述地点;

(d) 把所述移动对象的实体移动到所述地点；以及

(e) 通过使所述移动对象的实体说明所述载运智能体和指示所述具体活动执行作业，在所述地点执行所述具体活动的作业。

3．一种执行移动对象的方法，用于在网络上分布的至少一个地点上执行作业，该方法特征在于包括步骤:

(a) 把一个载运智能体移动到所述地点，该载运智能体持有定义着要在所述地点执行的一个作业;

(b) 把移动对象的实体移动到所述地点，移动对象含有说明所述载运智能体的信息并指示所述载运智能体在所述地点执行所述具体活动的作业; 以及

(c) 通过使所述移动对象的实体说明所述载运智能体和指示所述具体活动执行作业，在所述地点执行所述具体活动的作业。

4．一种存储至少一部分的移动对象的记录介质，这些移动对象在网络上分布的不同地点处执行不同的作业，该记录介质特征在于包括:

(a) 第一载运智能体，其持有定义着要在第一地点执行的某作业的第一具体活动;

(b) 第二载运智能体，其持有定义着要在第二地点执行的某作业的第二具体活动; 以及

(c) 移动对象的实体，它们含有说明所述第一载运智能体的信息及说明所述第二载运智能体的第二信息，并指示所述第一载运智能体在所述第一地点执行所述第一具体活动的作业，而且指示所述第二载运智能体在所述第二地点执行所述第二具体活动的作业。

5．一种存储移动对象的记录介质，这些移动对象用于在网络上分布的多个地点上执行某作业，该记录介质特征在于包括:

(a) 载运智能体，其持有定义着要在所述地点执行的作业的具体活动; 以及

(b) 移动对象的实体，它们含有说明所述载运智能体的信息并指示所述载运智能体在所述地点执行所述具体活动的作业。

6．一种存储至少一部分的移动对象的记录介质，移动对象在网络上分布的不同地点执行不同的作业，其特征在于记录介质包括

(a) 第一代理活动，其含有说明所述第一载运智能体的信息并指示所述第一载运智能体在第一地点执行第一具体活动的作业，所述第一载运智能体持有定义着要在所述第一地点执行的某作业的所述第一具体活动；

(b) 第二代理活动，其含有说明所述第二载运智能体的信息并指示所述第二载运智能体在第二地点执行第二具体活动的作业，所述第二载运智能体持有定义着要在所述第二地点执行的某作业的所述第二具体活动，以及

(c) 一个次序表，该次序表定义所述第一代理活动和所述第二代理的执行次序。

7．一种存储移动对象的记录介质，这些移动对象响应移动对象的实体的指令执行作业，移动对象的实体含有到达在网络上分布的某地点处的代理活动，该记录介质特征在于包括：

(a) 具体活动，其定义着要在所述地点执行的某作业；

(b) 提供给所述代理活动的具体活动表，以使所述代理活动说明所述具体活动；以及

(c) 一种用于从所述代理活动接收执行该作业的指令消息的方法。

8．一种存储移动对象的记录介质，这些移动对象响应移动对象的实体的指令执行作业，移动对象的实体含有到达在网络上分布的某地点的代理活动，该记录介质特征在于包括：

(a) 具体活动，其定义着要在所述地点执行的某作业；以及

(b) 一种用于从所述代理活动接收执行该作业的指令消息的方法。

说　　明　　书

执行移动对象的方法以及存储移动对象的记录介质

本发明涉及用于把移动智能体移动到网络上的远程服务器中并且在分布式计算机环境下的远程服务器中形成一条指令的移动智能体技术，本发明具体地涉及一种用于把移动智能体移动到远程服务器的系统。

已经存在用于把移动智能体移动到网络上的远程服务器中以在分布式计算机环境下形成一条指令的移动智能体技术(PUPA 7-182174 (USP5，603，031)，PUPA7-509799 (国际申请号：PCT/OS94/07397，国际公布号：WO97/02219)，Fumihiko Nishida、Susumu Fujiwara 等的“Latest Internet Technology, Nikkei Communication, separate volume” pp104-117, Nikkei BP Co.)。

这种移动智能体包括二种基本要素，即“移动智能体”和“地点(place)”。移动智能体可以在保持其内部状态下绕网络上存在的各地点移动。移动智能体可在一个地点上和其它的智能体(移动智能体或非移动智能体)接触以便接受必要的服务。“地点”是由智能体移动到的网络上已有服务器提供的地点，它支持智能体间的接触并消除硬件之间及平台之间的差异。

这样的移动智能体允许移动智能体执行迄今仍由人完成的工作，诸如根据雇员的时间或会议室的使用安排调整内部会议的安排以及获得分布在网络上的所需信息。

如由本申请的申请者于 1997 年 4 月 10 日归档的日本专利申请 9-92091(本申请归档时其仍待公开)中所公开的那样，如果提供用以定义移动介质的所需行为模式的模板(仲裁模板)、并且控制移动智能体作为前驱(前一地点处的活动作业)和后继(后一地点处的活动作业)漫游到的各地点上发出请求的活动，存在着把移动介质的复杂行为分类成基本行为模式技术。

借助这种技术，移动智能体绕各地点漫游和持有所有定义着要在每个地点中执行的工作的活动，并且产生这样的问题，即，当移动时信息量很大从而网络上的负载过重并且在每个地点地占用过多的存储区。

此外，因为仅对预定的协同算法(方案)应用模型，不能实现灵活的行为，诸如在释放移动智能体后的移动期间进行重新装配或增添要被执行的作业。

本发明的一个目的是提供一种在移动期间使网络负载为最小的移动智能体。

本发明的另一个目的是提供一种使各地点处所占据的资源为最小的移动智能体。

本发明的又一个目的是提供一种允许在移动期间灵活地修改要被执行的作业的移动智能体。

本发明的再一个目的是缩短从开始释放移动智能体到回送结果为止所需的处理时间。

本发明中，在载运智能体上载运用于移动到各个地点上的定义着要在各地点上执行的作业的具体活动。另一方面，移动对象的实体持有一个代理活动，该代理活动向载运智能体上载运的具体活动发送一个要被执行的指令。移动对象实体绕各地点循环并向移动到各地点的具体活动发送一条要执行的指令。

在本发明的一种形式中，提供一种执行移动对象的方法，以在网络上分布的不同地点上执行不同的作业，该方法包括步骤:

(a) 生成第一载运智能体，其持有定义着要在第一地点上执行的一项作业的第一具体活动;

(b) 生成第二载运智能体，其持有定义着要在第二地点上执行的一项作业的第二具体活动;

(c) 生成移动对象实体，其含有说明所述第一载运智能体的信息并含有说明所述第二载运智能体的第二信息，指示所述第一载运智能体在所述第一地点执行所述第一具体活动，并指示所述第二载运智能体在所述第二地点执行所述第二具体活动;

(d) 把所述第一载运智能体移动到所述第一地点;

(e) 把所述第二载运智能体移动到所述第二地点;

(f) 把所述移动对象的实体移动到所述第一地点;

(g) 通过使所述移动物象的实体说明所述第一载运智能体和指示所述第一具体活动执行作业，在所述第一地点执行所述第一具体活动的作业;

(h) 把所述移动对象的实体移动到所述第二地点; 以及

(i) 通过使所述移动对象的实体说明所述第二载运智能体和指示所述第二具体活动执行作业，在所述第二地点执行所述第二具体活动的作业。

在本发明的另一种形式中，提供一种执行移动对象的方法，以在网络上分布的至少一个地点处执行一项作业，该方法包括步骤:

(a) 生成一个载运智能体，其持有定义着要在所述地点执行的一项作业的具体活动;

(b) 生成移动对象的实体，其含有说明所述载运智能体的信息并指示所述载运智能体在所述地点执行所述具体活动;

(c) 把所述载运智能体移动到所述地点;

(d) 把所述移动对象的实体移动到所述地点; 以及

(e) 通过使所述移动对象的实体说明所述载运智能体和指示所述具体活动执行一项作业，在所述地点执行所述具体活动的作业。

在本发明的另一种形式中，提供一种执行移动对象的方法，以在网络上分布的至少一个地点处执行一项作业，该方法包括步骤:

(a) 把一个持有定义着要在所述地点执行的一项作业的具体活动的载运智能体移动到所述地点;

(b) 把移动对象的实体移动到所述地点，移动对象含有说明所述载运智能体的信息和指示所述载运智能体在所述地点执行所述具体活动的一项作业; 以及

(c) 通过使所述移动对象的实体说明所述载运智能体和指示所述具体活动执行该作业，在所述地点执行所述具体活动的作业。

在本发明的另一种形式中，提供一种记录介质，用于存储至少一部分

的移动对象，这些移动对象在网络上分布的不同地点上执行不同的作业，该记录介质包括:

(a) 第一载运介质，其持有定义着要在第一地点上执行的一项作业的第一具体活动;

(b) 第二载运介质，其持有定义着要在第二地点上执行的一项作业的第二具体活动; 以及

(c) 移动对象的实体，它们含有说明所述第一载运智能体的信息并含有说明所述第二载运智能体的第二信息，它们指示所述第一载运智能体在所述第一地点执行所述第一具体活动的作业，并且指示所述第二载运智能体在所述第二地点执行所述第二具体活动的作业。

在本发明的另一种形式中，提供一种存储移动对象的记录介质，用于在网络上所分布的各地点上执行一项作业，该记录介质包括:

(a) 载运智能体，其持有定义着要在所述地点处执行的一项作业的具体活动; 以及

(b) 移动对象的实体，它们含有说明所述载运智能体的信息并指示所述载运智能体在所述地点执行所述具体活动的一项作业。

在本发明的一种形式中，提供一种记录介质，用于存储至少一部分的移动对象，这些移动对象在网络上分布的不同地点上执行不同的作业，该记录介质包括:

(a) 第一代理活动，其含有说明所述第一载运智能体的信息，所述第一载运智能体持有定义着要在第一地点处执行的一项作业的第一具体活动，该第一代理活动还指示所述第一载运智能体在所述第一地点执行所述第一具体活动的作业;

(b) 第二代理活动，其含有说明所述第二载运智能体的信息，所述第二载运智能体持有定义着要在第二地点处执行的一项作业的第二具体活动，该第二代理活动还指示所述第二载运智能体在所述第二地点执行所述第二具体活动的作业; 以及

(c) 一个次序表，其定义执行所述第一代理活动及所述第二代理(活动)的次序。

在本发明的另一种形式中，提供一种记录介质，用于存储响应移动对象的实体的指令执行作业的移动对象，移动对象的实体包含着抵达到网络上分布的某地点的代理活动，该记录介质包括：

(a) 一个定义着要在所述地点上执行的一项作业的具体活动，

(b) 一个提供给所述代理活动的具体活动表，用于为所述代理活动说明所述具体活动，以及

(c) 一种用来接收一条指令消息以执行来自所述代理活动的作业的方法。

在本发明的另一种形式中，提供一种用于存储移动对象的记录介质，响应由包含着抵达到网络上分布的某地点的代理活动的移动对象的实体所提供的指令，这些移动对象执行作业，该记录介质包括：

(a) 一个定义着要在所述地点上执行的一项作业的具体活动，以及

(b) 一种用来接收一条指令消息以执行来自所述代理活动的作业的方法。

图 1 表示分布式网络环境，本发明生成的移动智能体在其中运行。

图 2 表示一种方式的例子，在该方式下本发明生成的移动智能体在分布式网络上移动。

图 3 是本发明的最佳实施方式中的移动智能体生成系统的硬件配置原理图。

图 4 是一个功能方块图，表示本发明的移动智能体生成系统中的处理部件的一种实施方式。

图 5 是从 GUI 方案定义数据生成部分观察时本发明的移动智能体生成系统中的处理部件的一种实施方式的功能方块图。

图 6 是一个功能方块图，表示在本发明的移动智能体生成系统生成的移动智能体的运行期间的处理部件的一种实施方式。

图 7 是本发明的最佳实施方式中生成的移动智能体的对象图。

图 8 是本发明的最佳实施方式中生成的移动智能体的对象图。

图 9 是本发明的最佳实施方式中生成的方案定义数据的对象图。

图 10 表示本发明的最佳实施方式中对象之间的消息流。

图 11 表示本发明的最佳实施方式中的移动智能体生成系统的用户接口。

图 12 表示本发明的最佳实施方式中的移动智能体生成系统的用户接口。

图 13 表示本发明的最佳实施方式中的移动智能体生成系统的用户接口。

图 14 表示本发明的最佳实施方式中的移动智能体生成系统的用户接口。

图 15 表示本发明的最佳实施方式中的移动智能体生成系统的用户接口。

图 16 表示本发明的最佳实施方式中的移动智能体生成系统的用户接口。

图 17 是一个流程图，表示本发明的最佳实施方式中的一个根据方案定义数据生成运行代码(方案)的过程。

图 18 是一个流程图，表示本发明的最佳实施方式中的一个根据方案定义数据生成运行代码(方案)的过程。

图 19 是一个流程图，表示本发明的最佳实施方式中的一个生成载运智能体并且传输到各地点的过程。

图 20 是一个流程图，表示本发明的最佳实施方式中的一个生成载运智能体并且传输到各地点的过程。

图 21 是一个流程图，表示本发明的最佳实施方式中的方案对象的执行进程。

图 22 是本发明的最佳实施方式中的仲裁模板的功能方块图。

图 23 是本发明的最佳实施方式中的一个地点的示意图。

图 24 是一个流程图，表示本发明的最佳实施方式中对仲裁模板的简要处理。

图 25 是一个示意图，表示一种状态，在该状态下本发明的最佳实施方式中的移动对象从起始地点移动到目的地点。

图 26 是一个示意图，表示一种状态，在该状态下本发明的最佳实施

方式中的移动对象从起始地点移动到目的地点。

图 27 是一个流程图，表示本发明的最佳实施方式中处理起点处的迭代（ Iteration ）模板的过程。

图 28 是一个示意图，表示一种状态，在该状态下本发明的最佳实施方式中的移动对象从起始地点移动到目的地点。

图 29 是一个示意图，表示一种状态，在该状态下本发明的最佳实施方式中的移动对象从起始地点移动到目的地点。

图 30 是一个流程图，表示本发明的最佳实施方式中处理起点处的与分离（ ANDSplit ）模板的过程。

图 31 是一个流程图，表示本发明的最佳实施方式中决定当前节点的逻辑。

图 32 是一个流程图，表示本发明的最佳实施方式中处理目的地点的与连接（ ANDJoin ）模板的过程。

图 33 是一个流程图，表示本发明的最佳实施方式中处理目的地点的与连接（ ANDJoin ）模板的过程。

图 34 表示一个进程概要，在该进程中本发明的最佳实施方式里的移动对象接收结果（ Result ）。

图 35 是一个示意图，表示一种状态，在该状态下本发明的最佳实施方式中的移动对象从起始地点移动到目的地点。

图 36 是一个流程图，表示本发明的最佳实施方式中处理起点处的或分离（ ORSplit ）模板的过程。

图 37 是一个示意图，表示一种状态，在该状态下本发明的最佳实施方式中的移动对象从起始地点移动到目的地点。

图 38 是一个流程图，表示本发明的最佳实施方式中处理目的地点的或连接（ ORJoin ）模板的过程。

参考数字的说明

100： 节点系统

201： GUI 方案节点库

203： GUI 仲裁模板

205： GUI 活动
207： GUI 方案定义数据
209： GUI 方案定义数据生成部分
211： 运行代码生成部分
213： 方案生成部分
215： 方案 ID 生成部分
221： 方案对象
222： 代理活动
223： 方案配置部分
224： 方案分离部分
225： 次序表
227： 方案 ID
229： 网络拓朴
230： 载运智能体
231： 输入事件获取部分
232： 目的地地点地址
233： 方案对象操作部分
234： 载运智能体 ID
236： 具体活动表
237： 方案对象属性修改部分
243： 方案显示部分
249： 运行代码生成部分
251： 仲裁智能体
253： 移动智能体生命周期控制部分
255： 方案执行部分
261： 方案对象
263： 方案执行部分
265： 次序表
267： 当前节点

269：具体活动表

270：移动对象的实体

271：仲裁模板

273：仲裁智能体控制部分

275：方案步骤执行部分

277：地址引用部分

280：载运智能体

281：代理活动

282：具体活动

283：代理活动执行部分

284：具体活动执行部分

285：消息子例程

286：消息子例程

291：作用智能体

293：消息子例程

295：消息处理部分

A．概述

现在参照各附图说明本发明的一种实施方式。图 1 表示一个执行根据本发明生成的移动对象的分布式网络环境 150。各服务器 112-117 配备有地点 102-107，在各个地点可对移动智能体 125、135、141 等提供服务。该分布式网络环境中的地点组称为一个群。

在客户系统 101 中存在一个用于生成移动智能体的移动智能体生成部分 113。移动智能体生成部分 113 向仲裁智能体 111 发送一个生成的方案，通过执行该方案该仲裁智能体 111 可移动到指定的地点。

移动智能体 125 等可以和每个地点 102-107 处存在的另一个智能体(和移动智能体接触以便提供服务的智能体具体地称为作用智能体)接触，可以发送请求并接收该请求的结果。地点支持智能体之间的接触。此外，移动智能体 125 等可以在结果 139、143、153、163 中持着从作用智能体接收的请求结果，继续移动并且可以在移动期间向结果施加各种

工作，例如结果的组合和分离。

在本发明中，通过仲裁模板 181-187 控制转动智能体的移动、分离和消除，用户可以通过组合处理方式所需的各种模板简单地定义包含着分离和合并的复杂工作，如图 2 中所示。

B．硬件配置

图 3 表示本发明的最佳实施方式中的移动智能体生成系统的硬件配置原理图。移动智能体生成系统 100 包含中央处理机(CPU)1 和存储器 4．通过总线 2， CPU1 和存储器 4 与作为辅助存储器的硬盘机 13 等连接。经过软盘控制器 19(或诸如 IDE 控制器 25、 SCSI 控制器 27 之类的控制器)，软盘机 20(或诸如 MO、 CD-ROM 等的介质驱动机 13、 26、 28、 29、 30)和总线 2 连接。

软盘(或诸如 MO、 CD-ROM 等的介质)被插入到软盘机 20(或诸如 MO、 CD-ROM 等的介质驱动机 26、 28、 29)中。可以在软盘中和硬盘机 13、 30 的记录介质中以及 ROM14 中记录一种计算机程序代码，该计算机程序代码向 CPU 发出指令以和操作系统合作实现本发明，通过装入到存储器 4 中可执行该程序代码。可压缩该计算机程序代码，或把该程序代码分段以便记录在多块介质上。

此外，移动智能体系统 100 可以是一个备有用户接口硬件的系统，用户接口硬件包括用于输入屏幕位置信息的指点器 7(鼠标、操纵杆、跟踪球等)，用于支持击键输入的键盘 6，以及用于向用户呈现图象数据的显示器 11、 12．扬声器 23 经放大器 22 接收来自声频控制器 21 的声频信号，以作为语音输出。

作为本发明的移动智能体生成系统 100 的一种输入的 GUI 方案节点库(后面说明)被存储在硬盘 30 中并且经过 SCSI 接口 27 输入到数据库检索系统。还有可能和其它的计算机等通信，以通过串行端口 15 和调制解调器或者诸如令牌环的通信适配器，访问其它系统的数据库或者访问在诸如软盘 24 等的记录介质上存在的数据库。

因此，容易理解本发明可以用常规个人计算机(PC)、工作站、在家用电器例如电视机和传真机中提供的计算机以及它们的组合来实现。但是请

注意，这些部件是按示范目的给出的，并不意味着所有这些部件都是本发明不可缺少的部件。具体地，由于本发明的目的是生成移动智能体，对于本发明的一种形式一些部件，其中包括串行端口 15、通信适配卡 18、声频控制器 21、放大器 22 和扬声器 23，不是不可缺少的。

对于操作系统，那些缺省下支持 GUI 多窗口环境的操作系统，例如 Windows(微软公司的商标)、OS/2(IBM 公司的商标)、AIX(IBM 公司的商标)上 XWINDOW 系统(MIT 的商标)，是合乎需要的，但是本发明使用的操作系统不限定在任何特定的操作系统环境下。

尽管图 3 示出独立环境下的系统，本发明可在客户/服务器系统中实施，在客户/服务器系统中客户机是通过以太网、令牌环等和服务器连接的 LAN，在服务器一侧配置下面描述的方案分离部分等而其余功能配置在客户侧。因此，在设计时要决定把什么功能放在服务器一侧以及把什么功能放在客户机一侧。多个计算机的组合的各种改进、这些计算机的功能的分配以及这些方面的实施方式属于本发明的原理之内。

C．系统配置

现参照图 4 和图 5 的方块图，说明本发明的最佳实施方式中的移动智能体生成系统的系统配置。

图 4 是一个功能方块图，表示移动智能体生成系统 210 中处理部件的一种实施方式。运行代码生成部分 241 包括方案生成部分 213 和方案 ID 生成部分 215。方案 ID 生成部分 215 对方案分配一个在网络上是唯一的方案 ID。在本发明的最佳实施方式中，方案 ID 生成部分 215 通过把互联网的 URL、端口号、智能体 ID(被分配给移动智能体生成部分)以及序列号(每当生成一个方案 ID 时移动智能体生成部分分配该号)组合起来生成方案 ID。

方案生成部分 213 生成仲裁智能体和基于 GUI 方案定义数据 209 的方案。所生成的方案包括方案配置部分 223 和次序表 225，方案配置部分 223 控制要运行的仲裁模板的信息及活动等，次序表 225 定义仲裁模板及活动的次序。方案配置部分 223 具有与方案节点对象(一个仲裁模板和一项活动的集合名称)有关的信息并产生网络拓朴 231。

GUI 方案定义数据生成部分 207 利用 GUI 方案节点库 201 中的信息，生成 GUI 方案定义数据 209 。

方案分离部分为各个地点生成载运智能体 230，该智能体移动到地点以便形成由运行代码生成部分 211 生成的一个方案。此外，方案分离部分用代理活动 222 代替和次序表关联的活动 205(称为具体活动以和下面说明的代理活动区分)。

代理活动 222 包含使具体活动 205 被执行的消息但不包含用于执行本发明的最佳实施方式中的实体对象的代码。载运智能体 230 被传送到各个目的地地点，等待移动对象 221 的实体的到达(与此相反，存在着移动对象 221 的实体等待载运智能体 230 到达的情况)并且响应移动对象的实体提供的指令执行具体活动 205 。

图 5 是一种从 GUI 方案定义数据生成部分 207 观察时，移动智能体生成部分 210 中的处理部件的实施方式的功能方块图。

向 GUI 方案定义数据生成部分 207 提供输入信息的输入事件获取部分 231 获取诸如来自用户的击键输入或鼠标器点击的事件，并把该事件变换成一条消息，该消息可由 GUI 方案定义数据生成部分 207 的方案对象操作部分 233 解释以发送给部分 233 。

方案对象操作部分 231 生成例如 GUI 活动或 GUI 模板的对象，并且根据事件的种类设定对象的属性。

方案对象属性修改部分 237 提供一个用来修改特定对象的属性的对话窗口。经过方案对象操作部分 231 把修改后的信息反映到 GUI 方案定义数据 239 。

GUI 方案定义数据保持部分 243 保持构成方案的活动、模板的逻辑信息以及图形图象信息。

方案显示部分 245 根据 GUI 方案定义数据 243 的图形图象信息在屏幕上显示方案的结构。

运行代码生成部分 241 生成运行代码 249(方案)，用于根据 GUI 方案定义数据的逻辑信息生成已定义的方案。

图 6 是在运行由移动智能体生成部分 210 生成的移动智能体 270 、

280 时的一种处理部件的实施方式的功能方块图。移动对象的实体 270 包括一个仲裁智能体 251、一个方案对象 261、一个仲裁模板 271 以及一个代理活动 281 。仲裁智能体 251 包括移动智能体生命周期控制部分 253 和方案执行部分 255。移动智能体生命周期控制部分 253 对移动对象的实体 270 执行移动、复制、擦除等。方案执行部分 255 识别到达新地点并且指示方案对象 261 执行方案。

另一方面，方案对象 261 包括方案执行控制部分 263 、当前节点 267 和次序表 265。一旦从仲裁智能体的方案执行部分接收到执行某方案的指令，方案执行控制部分 263 就参照当前节点 267 和次序表 265 确定当前要执行的方案节点对象 271 、 281 ，并且请求方案节点对象 271 、 281 执行。

当前节点 267 相当于后面要说明的方案节点迭代器并且监视当前在执行哪一个方案节点对象。次序表 265 是一个规定将要执行的方案节点对象 271 、 281 的表。当不出现分支等时，方案对象 271 、 281 请求方案节点按次序表 265 的次序执行。

仲裁模板 271 包括仲裁智能体控制部分 273 、方案步骤执行部分 275 和地址引用部分 277。仲裁智能体控制部分 273 请求仲裁智能体 251 的移动智能体生命周期控制部分 253 移动、复制和擦除移动对象的实体 270。根据仲裁模板的种类以及在前一地点或下一地点中执行的控制指令，各不相同地定义方案步骤执行部分 275。地址引用部分 277 的作用是从代理活动 281 获取地址并向仲裁智能体控制部分 273 通知目的地地址。

代理活动 281 包括代理活动执行部分 283 和消息子例程 285。代理活动执行部分 283 通过消息子例程 285 与载运智能体交互，以判定该载运智能体是否是具有对应于该代理活动的具体活动 282 的载运智能体 280 并且提示对应的具体活动 282 执行一项作业。具体活动表 269 是一个规定被执行的具体活动 282 的表。

具体活动 282 包括活动执行部分 284 和消息子例程 286。活动执行部分 284 具有存储要在移动智能体所环绕的各地点上执行的指令以及存储从每个地点接收到的结果的功能。消息子例程 286 具有与作用智能体 291

交换消息的功能。

作用智能体 291 是在地点上存在的另一种智能体，并且具有对具体活动 282 的请求提供服务的功能。作用智能体 291 也具有一个消息子例程 293 以便和活动交换消息，并且具有一个消息处理部分 295，用于确定请求的内容和提供服务。

虽然说明了图 4 至图 6 的各个功能块，这些功能块只是逻辑功能块，并不意味着每个框是用集成的硬件或软件实现的。它们可以用统一的或共享的硬件或者软件实现。此外，并不意味着图 4 至图 6 中所示的功能块是本发明必不可少的部件。例如，由于当前节点控制部分 267 可以直接控制当前节点并且可以在与连接“ ANDJoin ”处理中规定获取其它结果的仲裁智能体，次序表 265 不是必不可少的部件。另外可以通过把运行代码生成部分放置在对其发送 GUI 方案定义数据的另一个计算机上，来生成运行代码。

D．对象配置

D-1．移动智能体的对象配置

图 7 和图 8 是本发明的移动智能体 270 、 280 的对象图。类仲裁智能体 303 、类方案 305 、类方案 ID 以及类方案迭代器 307 各具有一种方法，而类方案节点 311 、类活动 313 、仲裁模板 315 以及模板 321 至 326(例如链)各具有数据和一种方法。

在图 7 和图 8 中，三角形 302 、 314 、 316 表示继承来自父类的数据和方法，圆点 312 表示存在多个子类。方案节点是一种抽象的类并且通过公用接口定义仲裁模板和活动二种类。从而在生成以及执行方案期间，方案可以类似地处理这二种类。

移动智能体 301 是当前在分布式计算机环境上的每个地点中设置的一个函数(地点类)，用户可以通过向该类发出一个标识创建移动智能体的 API，方便地创造一个移动智能体(仲裁智能体)。前面在现有技术的说明中所描述的 Java 库和 Telescript 的移动对象当前都支持这种函数。

在本发明的最佳实施方式中，提供下述方法以便用代理活动替换具体活动并生成载运智能体。

[表 1]

```
SetProxyActivity (index, agentid) {
// 交换活动
conAct = -plan.elementAt (index);
address = conAct.getdestination 0;
proxyAct = new ProxyActivity (sddress, agentid);
-plan.setPlanNode (proxyAct, index);
// 生成载运智能体
carrierAgent = context.getAglet (agentid)
carrierAgent.sendMessage ("store", conAct, index);
}
```

方案节点类具有一个前驱表和一个后继表并且控制多个方案节点之间的关系。方案节点类提供方法“ setPredecessor(设置前驱)”、“ setSuccessor(设置后继)，并且定义节点之间的关系。方案节点的具体的类可以辨别各种方法之间的关系。

仲裁模板具有区分起点(移动前的地点)和目的地(移动后的地点)的状态信息以便连接在不同地点上执行的活动。在本发明的最佳实施方式中，仲裁模板只能连接活动对象。仲裁模板备有多个其中包括着“ Chain ”(链)的模板，并且允许设置一种基本的使多个分离的仲裁智能体在规定的独立作业上工作的机制。

活动持有在其上执行作业的地点的地址信息。活动可以通过响应来自外部的诸如“ execute Activity(执行活动)”的指令指示“ abstract Activity ”(抽象活动)，执行一个内部持有的作业。代理活动具有载运智能体 ID 的信息并指示具体活动执行作业，该载运智能体 ID 规定持有对应的具体活动的一个载运智能体。

[表 2]

```
executeActivity {
   index = -plan.indexof (this)
  agent = context.getAglet (id)
```

```
table = agent.sendMessage ("gettable")
conAct = table.elementAt (index) ;
object result = conAct.executeActivity 0;
}
```

D-2． GUI 方案定义数据的对象配置

图 9 是本发明的最佳实施方式中的 GUI 方案定义数据 239 的对象图。类 GUI 活动 403、类 GUI 仲裁模板 413、类 GUI 地点 405 以及类链接 407 各具有数据和一个方法。

在本图中，三角形 302、314、316 表示从父类继承数据及方法，黑圈 312 表示存在多个子类。

GUI 活动 403 是一种按 GUI 编码程序上的一项活动显示的对象，它具有后继表和前驱表的数据以便规定一个链接着在显示器上显示时的坐标值(显示起始位置： X1， Y1 以及显示结束位置： X2， Y2)的对象。

类似地， GUI 仲裁模板 409 是一种按 GUI 编码程序上的一个仲裁模板显示的对象，它具有后继表和前驱表的数据以便规定一个链接着在显示器上显示时的坐标值(显示起始位置： X1， Y1 以及显示结束位置： X2， Y2)的对象。它还具有用来说明仲裁模板类型的模板类型数据。

链接对象 407 是一种用于显示 GUI 活动 403 和 GUI 仲裁模板 409 之间的链接关系的对象，它具有后继表和前驱表的数据以便规定一个链接着在显示器上显示时的坐标值(显示起始位置： X1， Y1 以及显示结束位置： X2， Y2)的对象。

GDI 地点 405 是一种用于显示在其上执行每个活动的地点的对象，它具有所包含的要被显示的 GUI 活动的坐标值信息(各 GUI 活动的显示起始位置： X1， Y1 以及显示结束位置： X2， Y2)的数据并具有该地点的地址信息。

D-3． 各种移动智能体的消息流

图 10 表示本发明最佳实施方式中所有类的主要消息流。仲裁智能体 351 向方案 353 发出“ nextStep ”(指示执行下一步骤的指令)，方案 353 对此响应向方案节点对象(仲裁模板 357 或代理活动 355)发出

“ nextStep ”指令。根据其类型或状态，仲裁模板 357 向代理活动 355 发送“ getARL ”(发送地址的指令)和“ isMarked ”(标记)。仲裁模板 357 还向仲裁智能体 351 发出“ clone ”(请求准备克隆)、“ dispatch ”(请求移动)和“ dispose ”(请求消除)。

代理活动 355 向载运智能体 352 发出“ getTable ”以得到具体活动表 354，它还利用所持有的作为关键字的一个索引(次序表的索引)从具体活动表 354 得到相应的具体活动 358。然后代理活动 355 向具体活动 358 发出“ executeActivity ”(请求执行活动)以使具体活动 358 执行用户规定的请求。

具体活动 358 对上述作出响应向作用智能体 359 发出“sendMessage”(请求发送消息)以便执行用户规定的请求。

E．准备方案定义数据

本发明提供一种使用户可简单地生成移动智能体的开发支持工具。图 11 至图 16 表示本发明的最佳实施方式中的该开发支持工具的用户接口。通过利用该开发支持工具，用户可以简单地生成上述方案定义部分的编码。

首先，当用户从下拉菜单中选择“ New ”(新方案)371 并用指点器点击时，方案对象操作部分 233 显示一个用来输入方案名的对话框 375。从而用户可以为该方案考虑一个名字，该名字不同于方案 ID 并且对用户来讲是熟悉的。用户接着向方案名的输入条目 376 输入方案名并且按下“ OK ”按钮以便打开定义方案的窗口 380。当选择下拉菜单中的“ Save ”(保存)372 或“ End ”(结束)374 时，可显示输入方案名的对话框 375。

接着用户利用指点器拖动活动框 420 的一个活动图标(ACT1)并放在用来定义方案的窗口 380 中。从而基于 GUI 方案节点库 201 中的原型数据生成 GUI 活动 403 以及 GUI 地点 405 的各个对象。调用所生成的 GUI 活动 403 的“ Display ”(显示)方法并在方案定义窗口 380 中显示 GUI 活动如图 12 中所示。在 GUI 活动 403 的前驱表以及后继表中设置“ O ”，以表示“不存在被链接的对象”。

如图9中所示，GUI活动431具有“Select”(选择)方法和“Move”(移动)方法，从而它能执行对应于用鼠标器点击“Select”而造成的处理并且能通过拖动鼠标修改显示位置。

用户接着利用指点器拖动模板框390中的“与分离(ANDSplit)”图标392并放在方案定义窗口380中。从而基于GUI方案节点库201的原型生成如图9所示的GUI仲裁模板410以及链接407的各个对象(生成2个链接对象)。调用所生成的GUI仲裁模板413的显示“Display”方法以及链接对象407的显示“Display”方法，以如图13中所示在方案定义窗口380中显示GUI仲裁模板433及链接对象432、434。

此刻，在链接对象432的后继表中设置GUI仲裁模板433的信息并在链接对象434的前驱表中设置GUI仲裁模板433的信息。另一方面，在仲裁模板433的前驱表中和后继表中分别设置规定链接对象432、434的信息。各链接对象的显示位置设置成根据GUI仲裁模板433的位置信息计算出的数值。

如图9中所示，GUI仲裁模板433还具有选择“Select”方法和“Move”方法，从而它能执行对应于用鼠标器点击“Select”而造成的处理并且能通过拖动鼠标修改显示位置。当移动GUI仲裁模板433时，根据移动量，更新由GUI仲裁模板433的前驱表和后继表所说明的对象的显示位置并且更新和这些对象链接着的一系列对象的显示位置，以便利用Display方法重画屏幕。

此后，当如图14中所示用户拖放链接对象432并且放在GUI活动431的上面时，方案对象操作部分233通过对GUI活动的显示位置和现在拖动的鼠标器的位置进行比较，规定被链接的GUI活动，并在GUI活动431的后继表中设置规定链接对象432的信息，以及在链接对象432的前驱表中设置规定GUI活动431的信息。方案对象操作部分233还把链接对象432的显示位置更新成一个相符于GUI活动431的显示位置的值，以便利用显示方法重画屏幕。

以这种方式，用户可以根据移动智能体所完成的作业的性质及过程，排列一个所需执行次序的活动，如图15中所示。此后，当用户通过保持

按下控制键并点击位置图标 399 选择希望在相同的地点上执行的一个所希望的活动时，显示出包含着被选活动的矩形 501 至 503 并且显示出地点属性设置窗口 510 。

在本发明的最佳实施方式中，通过沿 X 坐标轴对所选活的中心坐标进行排序，并计算其中心位于相邻活动的中心坐标的线段上的给定宽度的矩形，得到矩形 501 至 503 。顺便地说，可以用以上矩形代替其焦点位于活动的中心坐标上的椭圆。

当在本发明的最佳实施方式中的同一地点上规定多于 2 个的活动时，去掉各矩形的重迭部分以显示出单个多边形。此外，对代表 GUI 各地点的矩形或多边形赋予不同的彩色属性，从而操作者可以区分不同的地点。

虽然可以在按下键盘的控制键同时点击地点图标 399 来选择希望在同一地点上执行的活动，这也可以通过定点并拖动鼠标器以包围希望在同一地点上执行的活动来完成选择，也可以在选择后利用点击取消所做的选择。

当用户设置地点名或地点地址时，它们被设置成所选 GUI 活动 403 的 GUI 地点 405 的地址和地址名(图 9)。

F . 生成方案

图 17 是一个流程图，表示根据方案定义数据生成运行代码(方案)的过程。首先，生成方案对象(框 603). 方案生成通常包括生成方案 ID 的步骤以及建立一个与位于低于该方案对象的位置中一个对象的链接的步骤。

在本发明的最佳实施方式中，对方案对象分配一个在全局网络中为唯一的方案 ID 。在本发明的最佳实施方式中，方案 ID 生成部分 215(图 4)通过把互联网的 URL 、端口号、智能体 ID(分配给移动智能体生成部分)以及序列号(移动智能体生成部分每次生成方案 ID 时分配)组合起来生成方案 ID 。

在表 3 中示出方案 ID 生成部分的一个编码示例。在本发明的最佳实施方式中，向用户提供作为一个程序部分的方案 ID 生成部分的各指令。

[表 3]

```
public class Planner extends StationaryAgent{
    private Plan _plan;
    private PlannerFrame f;
    private URL _home;
    private AgletIdentifier _id;
    private int _planIndex = 0;

    public void onCreation (Object args){
        //生成一个方案选择窗口
        f = new PlannerFrame ("JMT -PlannerAgent", this);
        f.show();
        try{
        //获取存在移动智能体生成部分的 URL
         _home = getAgletContext 0.getHostingURL 0;
        //获取移动智能体生成部分的 ID
         _id = getIdentifier();
        }catch (InvalidAgletException e){

            Utility.print (this, e.getMessage ());
        }
}
```

在本发明的最佳实施方式中，用来生成方案对象的“ New Operator ”是以一个 Java 函数的形式提供的，通过把方案 ID 作为一个参数执行该类生成新的方案对象。

如前面所述，仲裁智能体是通过利用当前在各地点处提供的函数(地点类)产生的。在本发明的最佳实施方式中，通过把指示生成移动智能体的 API 施加到 Java 库的移动智能体的 Aglets 类生成仲裁智能体。

接着，生成一个和顶层 GUI 活动对应的活动(框 605)。具体地，根据

该 GUI 活动的活动类型获得运行时期类名(本情况下为 A1)，然后把类名“ A1 ”作为关键字生成其实例。把生成的活动增添到方案对象上(框 607)。

产生一个 FIFO(先进先出)队列以存储处理中的 GUI 对象(GUI 活动、 GUI 仲裁模板或链接对象)(框 609)并把顶层 GUI 对象(GUI 活动)放在该队列中(块 611)。通过查看前驱表是否为空等判定顶层 GUI 对象。

然后从生成的队列中取出一个元素(GUI 对象)以执行一个子程序(框 613)，该子程序增添后继表里存在的一个活动或一个仲裁模板，如图 18 中所示。

图 18 是一个表示该子程序的处理过程的流程图。首先，把后继表的元素数量和值 i 进行比较以判定是否要对后继表中的所有元素执行这种处理(框 665)。

当后继表的元素数量 > i 时，取出后继表的第 i 个 GUI 对象(框 657)。

接着判定该取出的 GUI 对象是否已经存在在该队列中(框 658)，仅当不存在时才把该元素放入到在图 17 的框 609 中生成的队列中(框 609)。

然后判定该取出的元素(GUI 对象)是否是 GUI 活动或 GUI 仲裁模板(块 660)。若取出的元素是 GUI 活动或是 GUI 仲裁模板，则生成对应的活动或仲裁模板(框 661)。此刻，若它是一个 GUI 活动，则获取对应 GUI 地点的地址信息。

接着从当前 GUI 对象的前驱表的链接对象获取前驱索引，以便设置它自己的前驱表。在除顶层之外的 GUI 活动或 GUI 仲裁模板里，先驱表必定包含链接对象的信息。由于在顶层 GUI 活动的先驱表中设置空信息，在其自己的前驱表中也设置空。

然后，把生成的元素(活动或仲裁模板)增添到方案，并且在为该当前 GUI 对象的后继的链接对象中设置该方案中的次序表的索引。该链接对象把该值保持为前驱索引，该前驱索引被用作为用来产生该活动或该仲裁模板的前驱表的信息。

若所取出的元素不是 GUI 活动或不是 GUI 仲裁模板，则不生成对应的活动或仲裁模板而处理下一个元素。

在和后继表的元素数量相等的次数下重复框 657 至 665 的处理后，流程返回到图 17 以判定在该队列中是否存在元素。若存在元素，则取出元素(框 612)并重新执行图 18 所示的子程序以增添后继表中存在的活动或仲裁模板。

G．生成载运智能体、用代理智能体替换以及向各地点传输载运智能体

在生成一个方案后，启动如图 4 中所示的方案分离部分 224，以便为每个目的地地点生成载运智能体 230 并且用代理活动 222 替换与次序表相关的具体活动 205。并且把生成的载运智能体传送到各个目的地地点。

图 19 和图 20 是流程图，表示方案分离部分 224 以及载运智能体 230 进行的处理过程，以便生成载运智能体、用代理智能体替换并把载运智能体传输到各地点。

当启动该过程时(框 731)，首先初始化“i”(框 732)并从该方案中取出第 i 个元素(框(733)。判定所取出的元素是否是一个活动(具体活动)(框 734)。若取出的元素不是一个活动，则取下一个元素(块 735、733)。

虽然在本发明的最佳实施方式中用代理活动替换所有的具体活动并且在一个载运智能体上载运各具体活动以传送到各地点上，通过修改框 734 来判定元素是否是特定的具体活动可以实现用一个代理活动替代部分具体活动。

若取出的元素是一个活动，则生成一个代理活动(框 736)并设置从该具体活动中取出的目的地地点的地址(框 737)。然后查阅一个字表以判定是否存在一个具有相同目的地地点地址的载运智能体。

当不存在一个具有相同目的地地点地址的载运智能体时，生成一个载运该具体活动的载运智能体(框 739)并且在该生成的载运智能体中设置一个智能体 ID 以及该目的地地点地址(框 740)。顺便地说，载运智能体 ID 是根据方案 ID 和对应于该地点的一个序列号生成的。

然后，在工作表中设置该载运智能体的 ID 以及该目的地地点地址。并且，在该载运智能体的具体活动表的第 i 个位置中设置第 i 个具体活动(框 742)。在该载运智能体中可以设置一个时间炸弹并且在预定的时刻熄灭。

当存在过去曾用来保持相同的具体活动并被发送到相同的地点上的某载运智能体(这可以通过在给定的时间间隔内保持一个工作表得到确定)时，可以用索引信息替换智能体 ID，这样可在不必生成载运智能体下命令规定并执行一个和过去曾发送到某代理智能体的具体活动相同的某载运智能体的具体活动。

当存在具有相同目的地地点地址的某载运智能体时，设置该载运智能体的具体活动表的第 i 项(框 742)。

另一方面，在代理活动中设置该载运智能体的 ID，而该代理活动又被设置(替换)到次序表的第 i 项中。通过对所有方案的各元素执行从框 733 到框 744 的处理，完成了对应于每个目的地地点的载运智能体的生成并且完成用代理智能体替换移动对象的实体的具体活动(框 745， 746)。

然后，把生成的各载运智能体发送到对应的目的地地点(框 748)。

上面所说明的示例是一种实施方式，其一旦生成持有具体活动的方案就用代理活动替换具体活动，这种实施方式是对另一种实施方式的改进，后一种实施方式中在直接持有具体活动下移动对象的实体被移动。然而，可以在生成移动对象的实体时生成直接持有代理活动的方案。本发明的思想也包括着这样的实施方式。

H．执行方案

H-1．包括着链模板的拓扑情况

现在参照图 21 至 25 说明带有“ Chain ”（链）模板拓扑情况下的移动对象的操作。图 21 是表示方案执行过程的流程图，而图 22 是仲裁模板的功能方块图。图 25 是一个状态示意图，在该状态下移动对象从起点位置移动到目的地地点。首先参照图 21 。当生成仲裁智能体 31 并且分配方案 353 时(框 843)，该仲裁智能体自动地执行该方案(框 845)。

响应从仲裁智能体 351 发送给方案对象 353 的“ Nextstep ”(下个步骤)指令，方案对象 353 首先取出当前方案节点(块 847)。在本发明的最佳实施方式中，响应方案对象 353 的询问，保存哪一步方案节点中的方案迭代对象是当前对象的控制，并提供哪一步骤是当前步骤的信息(当前)。

当方案迭代器对象 352 响应当前的查询回送“ 1 ”(第一步)时，该方

案对象 353 取出“ act1 ”并接着要求“ act1 ”具有为在其中执行该请求而定义的活动，该请求由该活动控制(框 849)。

该活动判定其中所持有的智能体类名(智能体名)是否存在于该相同的地点。如图 23 中所示，仲裁智能体 350 一旦到达地点 360 便通知地点 360 它的智能体名，地点 360 利用智能体表 369 控制智能体名。因为地点具有回送“ Agent List ”(智能体表)的函数，地点 360 处存在的智能体可以通过对地点 360 发出“ get AgentList ”(得到智能体表)指令提出查询来寻找相同地点处存在的某智能体。

由于“ act1 ”已被一个代理活动替换，它判定是否存在一个和其持有的载运智能体 ID 相对应的智能体，若不存在该智能体，则等待载运智能体的抵达(进入休眠方式)。

当存在对应的智能体时，“ act1 ”指示该载运智能体执行具体活动表的对应索引(和该代理活动相关的次序表号)的活动。

对此响应，具体活动根据内部持有的消息的内容搜索一个作用智能体。该具体活动把其中持有的一个消息(请求)发送到所找出的智能体(作用智能体)361 。该作用智能体 361 响应该请求又对提出请求的具体活动 358 发送一个回送对象。该具体活动 358 再向提出请求的代理活动发送所接收的对象执行指令以作为其回送。该代理活动把所接收的对象存储到“ Result ”的记录 525 中。

该代理活动通知方案对象 353 已完成所请求的作业，并且方案对象 353 标记“ act1 ”以表明该过程已完成(框 855)。接着方案对象 353 要求方案迭代器 352 向前移动步骤(框 859)，从而方案迭代器 352 向方案对象 353 提供当前信息。

因为当前值为“ 1 ”(第二步)，该方案对象取出“ Chain ”(链)模板(框 845 、 847)对象并要求“ Chain ”执行(块 849)。

“ Chain ”的仲裁控制管理器 603 查询当前状态的状态控制部分 601 。由于缺省时状态控制部分 601 处于起点状态，它通知仲裁控制管理器 603 它是起点。一旦接收到来自状态控制部分 601 的状态信息，仲裁控制管理器 603 就要求状态控制部分切换状态。状态控制部分 601 对此作出

响应把状态从起点切换到目的地。

图 24 中示出模板完成的状态切换以及起点处或目的地处的处理流程。该流程图表示对于多种模板为相同的操作，但是框 825 中的起点处以及框 831 中的目的地处的处理内容是随不同类型的模板变化的。

仲裁控制管理器 603 要求起点仲裁模块 605 执行。在起点仲裁模块 605 以及目的地仲裁模块 607 中控制一组由模板在各地点上执行的指令。

起点仲裁模块605通过查询仲裁智能体引用模块599获得当前仲裁智能体 351 所存在的地点的地址。

起点仲裁模块 605 还要求后继表 609 发送在其中登记的活动的地址。后继表 609 从登记的活动(“ act2 ”)得到地址并把该地址发送到起点仲裁模块 605 。

起点仲裁模块 605 把该活动的地址和当前仲裁模板存在着的地址进行比较，若地址不匹配则把该仲裁智能体移动到该活动的起址。具体地，起点仲裁模块 605 发出一条命令经过仲裁智能体引用模块进行移动的指令。若地址匹配，该处理结束。

方案对象再次要求方案对象在目的地处执行步骤 1 。类似地，它取出当前方案节点。该方案对象要求“ Chain ”执行下一步骤。“ Chain ”确认状态信息并执行目的地的处理。把状态信息恢复成起点状态。

当方案确认完成链“ Chain ”的目的地处理时，它取出下一个当前方案节点(“ act2 ”)并要求执行该活动。“ act2 ”获取“ act1 ”内部保存的结果。

在本发明的最佳实施中，目的地模块 607 查阅前驱表 591 ，并且发出要求“ Chain ”发送处理结果的“ getResult ”(取结果)的指令。对此响应，“ Chain ”搜索它保持的结果“ Result ”。但是，由于“ Result ”并不持有“ Result ”，它转向在前驱表 591 中登记的对象(“ act1 ”)以获取“ Result ”并且进一步查阅前驱表 591 把结果“ Resalt ”回送给“ act2 ”。当在前驱表 591 中不存在登记的对象时，向“ act2 ”发送“ Dummy ”(哑元)以指出不持有“ Result ”。备择地，这也可以通过在模板中准备一个存储“ Result ”的记录来实现。

“ act2 ”利用内部持有的智能体的类名作为关键字得到存在于同一地点上的一个报告智能体的指针。“ act2 ”向该报告智能体发送一个内部持有结果的消息，该结果是从作为参数“ act1 ”得到的。方案取出下一个当前节点。在本情况下，由于不存在当前节点，该方案检测出这种情况并终止处理。目的地仲裁模块 607 经过仲裁智能体引用模块 599 向该仲裁智能体发出一条取消该仲裁智能体的指令。

H-2．包含着迭代“ Iteration ”模板的拓扑情况

图 26 表示由“ Iteration ”(迭代)模板控制的移动对象的操作概要。观参照图 22 解释包含着迭代模板的拓扑情况下的处理。类似于链“ Chain ”的情况，生成仲裁智能体 351，然后发送一个作为参数准备的方案对象 353，并一步一步地执行方案 353。类似于“ Chain ”在起点处执行处理的情况，该方案对象 353 也取出当前方案节点(Chain)。同样，方案对象 353 在目的地处和活动“ act1 ”处执行处理，如在“ H-1．包含着链模板的拓扑情况”一节中所说明的那样。

当处理完“ act1 ”后，方案 353 接着要求迭代“ Iteration ”执行下一步骤。图 27 是一个流程图，表示在“ Iteration ”模板的起点处的处理过程。在“ Iteration ”模板起点处的处理中，它确定能否处理下一步骤，例如，“ act1 ”的结果信息是否与预定条件匹配(框 873)。

若满足该条件，则在后继表中存储内部持有的后继信息(框 875)。从该后继表的“ act2 ”中取出地址(框 877)并把仲裁智能体移动到该地址(框 879)。当不满足该条件时，则把状态设置成起点(框 881)。从回送的节点索引获得该方案节点的指针，在该方案节点中替换后继表的元素，并且流程进行到下一步骤(框 883)。实质上，如“ H-1．包含着“ Chain ”模板的拓扑情况”一节中所说明那样，“ act2 ”被处理并且完成整个处理。

H-3．包含着与分离“ ANDSplit ”模板和与连接“ ANDJoin ”模板的拓扑情况

图 28 和图 29 是由与分离模板及与连接模板控制的移动对象的操作概要。下面参照图 22 说明包含着与分离模板及与连接模板的拓扑情况。类似于链“ Chain ”情况，生成仲裁智能体 351，接着传送作为参数准备

好的方案对象353，并且一步一步地执行方案353。

以类似于“Chain”情况的方式，方案对象353取出与分离，后者是一个当前方案节点。方案353要求与分离执行下一步骤。图30是一个流程图，表示在与分离模板中处理起点的过程。

仲裁控制管理器603搜索状态控制部分601的状态信息，并获得缺省值“起点”信息。仲裁控制管理器603要求起点仲裁模块605在起点处处理。仲裁控制管理器603要求状态控制部分把状态改变成目的地(框893)。与分离参照后继表取出后继表的第一个活动(框897)并且在标记索引595中设置“0”(框899)。

接着仲裁控制管理器603要求仲裁智能体351经过仲裁智能体引用模块599准备该仲裁智能体的一份拷贝(框901)。响应这个请求的仲裁智能体还准备所有对象，其中包括该仲裁智能体、该仲裁智能体持有的一个方案、一个模板及一个活动的拷贝。

由复制准备出的拷贝的仲裁智能体把它自己的智能体ID通知到仲裁智能体引用模块599，以对完成拷贝的准备做出响应。仲裁智能体引用模块599通知起点仲裁模块605已完成拷贝的准备。

对此响应的起点仲裁模块605获取对应于来自后继表609的标记索引值(0)的那个活动(“act1”)的地址。起点仲裁模块605要求仲裁智能体351经过仲裁智能体引用模块599把拷贝移动到和“act1”的地址对应的地点上(框903)。

起点仲裁模块605参照后继表609判定是否登记着下个活动。若登记，则起点仲裁模块605通过类似的过程准备另一个拷贝并把它移动到“act2”的地点(框897-903)。此时，与移动到“act1”的拷贝相反，移动到“act2”的拷贝的标记索引595保持为“1”。

在处理了要移动到“act2”的拷贝的准备及移动后，起点仲裁模块605参照后继表609判定是否登记着下一个活动(框897)。由于在本例中后继表609里不存在登记的活动，起点仲裁模块605辨别出已完成对克隆的准备及移动的处理。

对此作出响应，起点仲裁模块605发出经过仲裁智能体引用模块599

擦除保持在仲裁智能体351的原始地点(起始地点)处的仲裁智能体351(框905)的指令。

另一方面，克隆的仲裁智能体在各自的地点上执行下个方案步骤。此刻，再次取出与分离，以作为当前节点。在图31中示出决定下个当前节点的逻辑的流程图。仲裁控制管理器603参照状态控制部分601获取状态信息，识别出现行状态是目的地，并且要求目的地仲裁模块607执行处理。

仲裁控制管理器603把状态信息恢复成起点状态。当方案确认完成对与分离的目的地的处理时，它取出和标记索引(“act1”或“act2”)对应的方案节点并且要求执行该活动。

方案对象353参照方案迭代器352取出下一个当前节点。此时，方案迭代器参照后继表609以及标记索引595确定登记值(图31)。

“act1”和“act2”获取由前一执行的活动持着的结果。在本例中，由于在该“ANDSplit”模板之前未执行过任何活动，所以“act1”和“act2”最终不能得到由前一执行的活动保持的结果。

最初在和“act1”对应的克隆智能体中把与连接选择为下一当前节点。方案对象353要求与连接模板的仲裁控制管理器603执行下个步骤。仲裁控制管理器603从状态控制部分601的信息辨别出应处理当前起点，并对起点仲裁模块605发出执行请求。它还要求状态控制部分601改变状态。

起点仲裁模块605参照后继表609获取在后继表609中登记的活动(“act3”)的地址信息，并且指示仲裁智能体351经仲裁智能体引用模块599移动到和所获取的地址信息相对应的地点。

对应于“act2”的克隆智能体也按类似于“act1”的方式处理，并且移动到“act3”的地点。由于这二个仲裁智能体到达新地点，所以它们分别执行下一步骤。方案对象353取出与连接模板作为当前节点并执行下个步骤。

图32和33是流程图，表示与连接模板目的地处的处理过程。与连接模板的仲裁控制管理器603参照状态控制部分601判定当前状态是目的

地，并要求目的地仲裁模块 607 执行。

每个仲裁控制管理器 603 把状态恢复到起点状态。和“ act1 ”对应的克隆的目的地仲裁模块 607 参照前驱表 591 取出标记成已完成的活动(本情况下为“ act1 ”)(框 913)。它判定所取出的活动是否具有前驱索引表中的最小下标(框 915)，并且若判定出它具有下标索引则接收其它的克隆智能体的结果。

具体地，目的地仲裁模块 607 要求同一地点智能体引用模块 597 获取同一地点处存在的智能体表。对此作出响应，同一地点智能体引用模块 597 对地点 360 发出“ getAgentList ”指令以获取智能体表。该地点回送“ AgentList ”以得到该同一地点处存在的智能体表。

同一地点的智能体引用模块 597 把该表发送给目的地仲裁模块 607 。目的地仲裁模块 607 从该表中查找具有相同方案 ID 的智能体框 921 。

当找到具有相同方案 ID 的某智能体时，获取该智能体的方案。接着确定所获取的方案的当前节点(框 929)，并且若它是相同的与连接接收该智能体的结果。在本发明的最佳实施方式中，还要判定当前节点索引是否相同(框 931 ， 933)，并且仅当相同时才进入合并处理。这样做是为了防止与连接要处理的主题未被辨别。目的地仲裁模块 607 参照前驱表查出存在多少个前驱，并且接收和前驱数量相同的结果。

具体地，目的地仲裁模块 607 参照接收到的方案的前驱表 591 ，取出标记成已完成的一个活动并且接收该活动持有的结果。把该结果设置成对应活动的结果。方案节点本身可能被替换(框 935)。图 34 表示接收结果的处理的概要。

当目的地仲裁模块 607 检测出完成对结果的设置时，它向对应的仲裁智能体 351 发出一条指示经过仲裁智能体引用模块擦除的指令(框 937)。接收到该指令的对应仲裁智能体擦除该仲裁智能体以及它所控制的对象(方案、模板、活动等)。

另一方面，和“ act2 ”对应的智能体到达新的地点后立即执行下个步骤。该智能体的目的地仲裁模块 607 还参照前驱表 591 取出标记着已完成的活动(该情况中为“ act2 ”)(框 913)。它判定所取出的活动是否具有

前继索引表中的最小下标(框 915)，并且确定它不是最小的下标。在本情况下，进入休眠状态以等待发送其它克隆智能体的结果(框 925)。

回到和“ act1 ”对应的克隆，目的地仲裁模块 607 接收数量和前驱表的数量相同的结果(前驱表的元素数量-1)，并且方案取出下一个方案节点(“ act3 ”)和在完成克隆的擦除后要求执行该活动。

“ act3 ”获取“ act1 ”和“ act2 ”内部持有的结果。在本发明的最佳实施方式中，目的地仲裁模块 607 向“ act1 ”及“ act2 ”(和该“ act1 ”对应的克隆)发出指令“ getResult ”(取结果)以要求发送处理结果。对此作出响应，“ act1 ”和“ act2 ”发送它们持有的结果。

“ act3 ”利用作为关键字的内部持有智能体的类名得到在同一地点处存在的某智能体的指针。“ act3 ”向报告智能体发送带有结果的内部持有消息，该结果是从作为参数的“ act1 ”、“ act2 ”处得到的。

该方案取出下个当前节点。由于本情况中不存在这样的当前节点，该方案查出这种情况并终止处理。仲裁智能体检测出处理的终止，并且擦除它控制的仲裁智能体及各对象以结束处理。

H-4．包含着或分离模板的拓扑情况

图 35 表示由或分离模板控制的移动对象的操作概要。在或分离情况中，对起点的处理如图 36 中所示，其与链和与分离情况不同。或分离模板具有一个前驱活动和多个后继活动。在起点地点，通过调用的“ choose Activity ”(选择活动)方法从后继表选择一个活动，并仅执行该选出的活动。

在处理或分离的地点中，首先把状态改变成目的地(框 982)。接着调用函数“ choose Activity ”，以选择一个要被执行的满足选出的预定条件的活动(框 983)。该活动的索引变成一个标记索引(框 984)。获取该选出活动的地址并请求把仲裁智能体 351 移动到该地址(框 985)。

H-5．包含着或连接模板的拓扑情况

图 37 表示或连接模板控制的移动对象的操作概要。在“ ORJoin ”情况下，对目的地的处理如图 38 中所示，其与链和与连接的情况不同。或连接模板具有多个前驱活动和一个后继活动。在目的地处，或连接执行

一个处理，在该处理中首先抵达的移动智能体生成一个“清除”(Dispose)智能体并擦除其它移动智能体。

首先到达目的地 690 的移动对象的或连接模板 694 发出“ findDisposeAgent ”方法，以判定是否在同一地点存在一个清除智能体(框 943)。若不存在，则生成一个擦除其它智能体的清除智能体(框 945)。此刻，以参数方式传送方案以及前驱表中的活动数目-1 。清除智能体等待前驱表中的活动数目-1 个移动对象以便擦除该对象，并且在履行其作用之后擦除自己。虽然在本发明的最佳实施方式中生存条件取决于一个对象是否首先到达目的地 690，结果的内容可以是继续进行处理的条件。

另一方面，其它移动对象的或连接模板 696 类似地发送“ findDisposeAgent ”方法以判定在同一地点上是否存在清除智能体(框 943)。由于本情况中已存在清除智能体，所以进入一个由清除智能体擦除的进程。具体地，该进程判定该清除智能体是否持有相同的方案(框 947)或持有相同的当前节点索引(框 949)，以便确认它是要被擦除的清除智能体。

当确认该清除智能体是应该擦除的时，向该清除智能体发送“ Countdown(递减计数)”消息以递减 清除智能体的处理次数（框 951)。接着，或连接模板 696 向仲裁智能体 692 发送一个除去请求以消除它。

虽然本发明的最佳实施方式还提供部分连接“ Join ”模板，这种模板具有多个前驱活动以及其数目少于这些前驱活动的多个后继活动，但省略掉对这种模板的说明，因为这种模板可以用前面说明的模板的组合实现。例如可以通过把多个前驱活动、一个链接某个后继活动的或连接模板、一个用于确定已执行所需数量的或连接模板的活动、一个根据较后的活动是否清除所需的条件的判定向该或连接模板回送的迭代模板、一个“哑元”活动以及一个分离到所需数量的与连接组合起来实现这种部分的连接模板。

如上面所说明，本发明在不需要高级编程知识下允许程序员用减小的开发量和开发时间开发移动智能体。

并且，可以尽可能多地减少开发移动智能体所需的开发量和开发时间。

此外，本发明可以提供一种用户友好的开发移动智能体的环境，该环境允许开发者直觉地领会其功能。

说 明 书 附 图

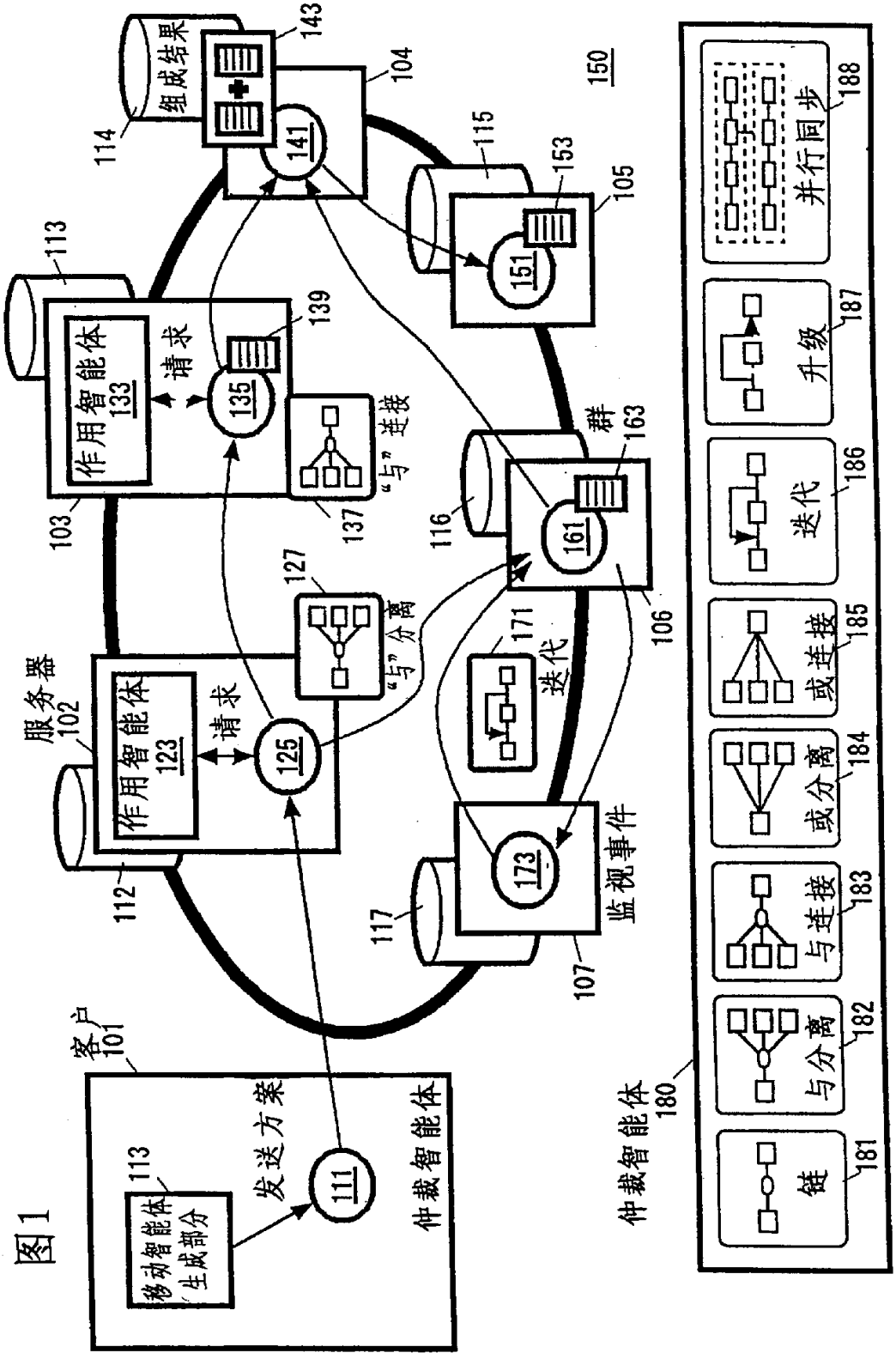

客户
101
移动智能体
生成部分
113
发送方案
111
仲裁智能体
服务器
102
作用智能体
123
请求
125
"与"分离
127
作用智能体
133
请求
135
"与"连接
137
139
组成结果
114
141
143
104
115
151
153
105
群
163
161
迭代
171
106
监视事件
173
107
117
112
103
116
113
150
仲裁智能体
180
链
181
与分离
182
与连接
183
或分离
184
或连接
185
迭代
186
升级
187
并行同步
188

图 1

图2

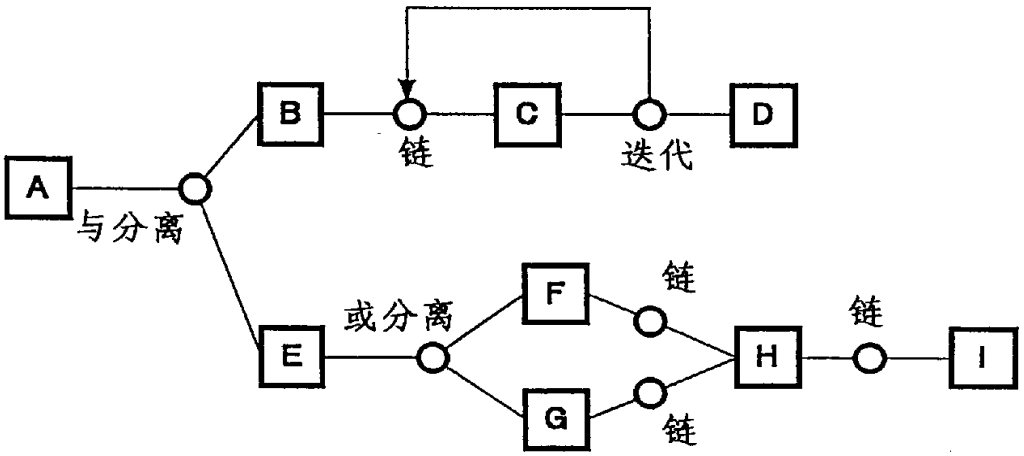


图 3

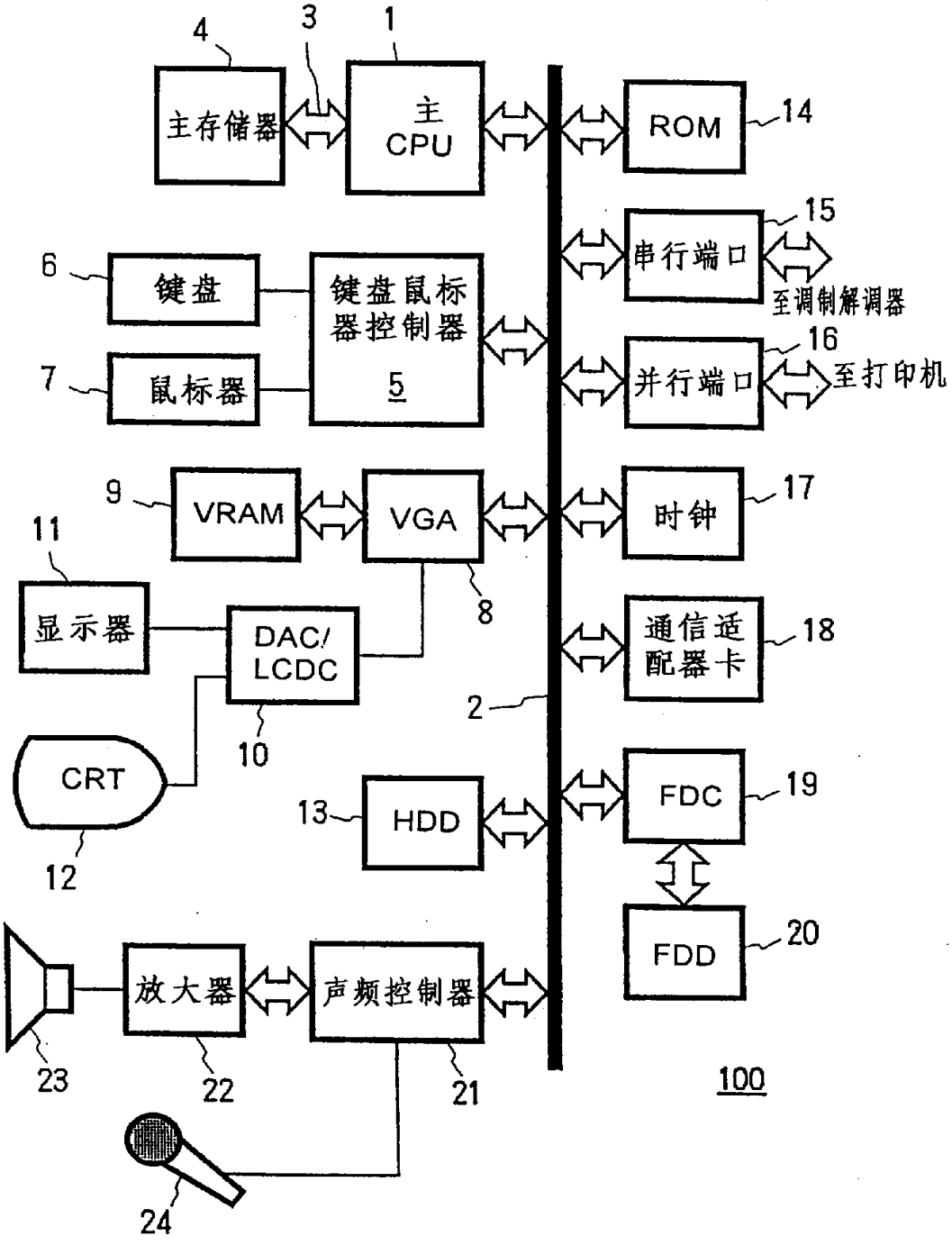

4
3
1
主存储器
主
CPU
ROM
14
15
串行端口
至调制解调器
6
键盘
键盘鼠标
器控制器
5
16
7
鼠标器
并行端口
至打印机
9
VRAM
VGA
时钟
17
11
显示器
DAC/
LCDC
8
通信适
配器卡
18
2
10
CRT
FDC
19
13
HDD
12
FDD
20
放大器
声频控制器
23
22
21
100
24

图4

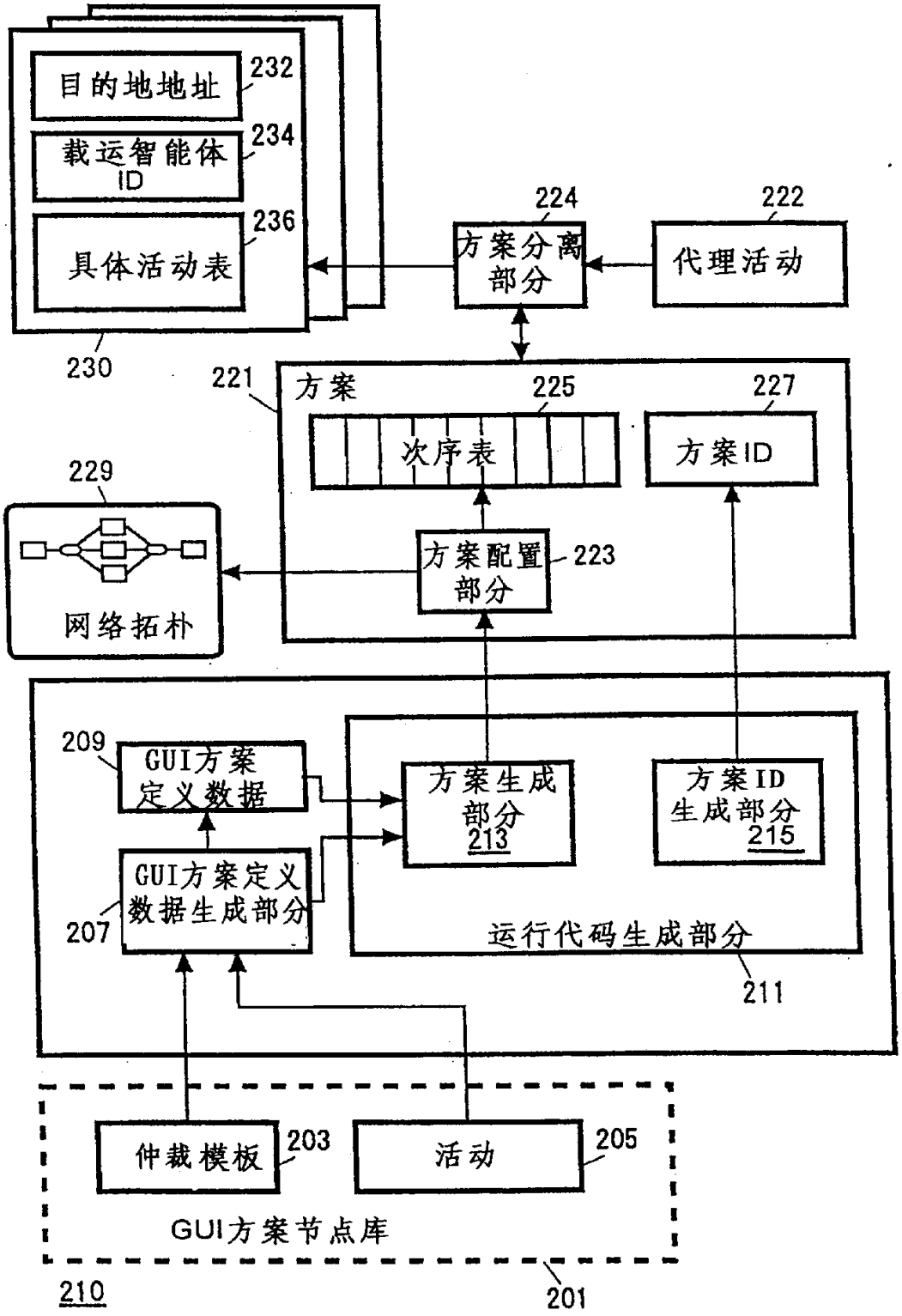

目的地地址
232
载运智能体ID
234
具体活动表
236
230
224
方案分离部分
222
代理活动
221
方案
225
次序表
227
方案ID
229
网络拓朴
方案配置部分
223
209
GUI方案定义数据
方案生成部分
213
方案ID生成部分
215
GUI方案定义数据生成部分
207
运行代码生成部分
211
仲裁模板
203
活动
205
GUI方案节点库
210
201

图5

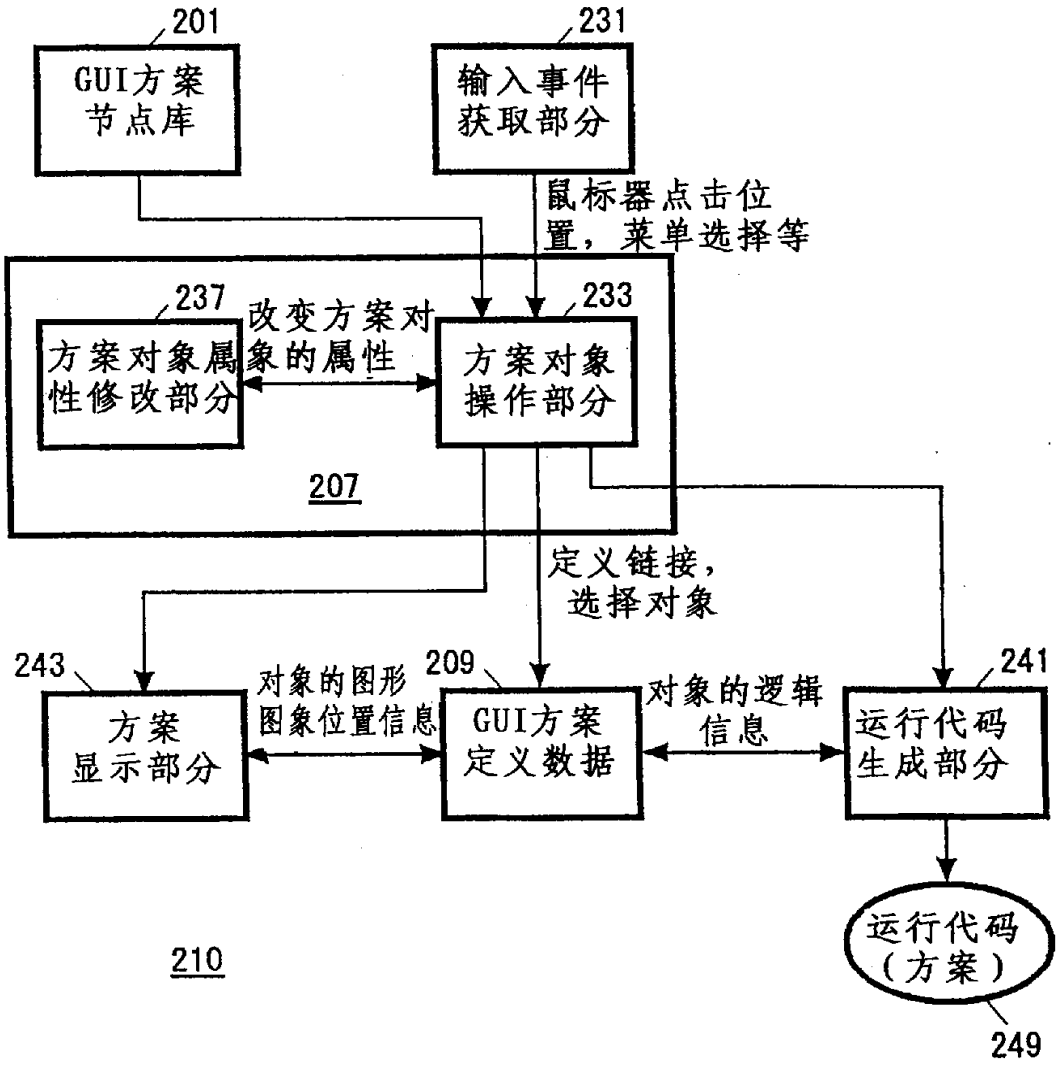

201
GUI方案节点库
231
输入事件获取部分
鼠标器点击位置，菜单选择等
237
方案对象属性修改部分
改变方案对象的属性
233
方案对象操作部分
207
定义链接，选择对象
243
方案显示部分
对象的图形图象位置信息
209
GUI方案定义数据
对象的逻辑信息
241
运行代码生成部分
运行代码（方案）
249
210

图 6

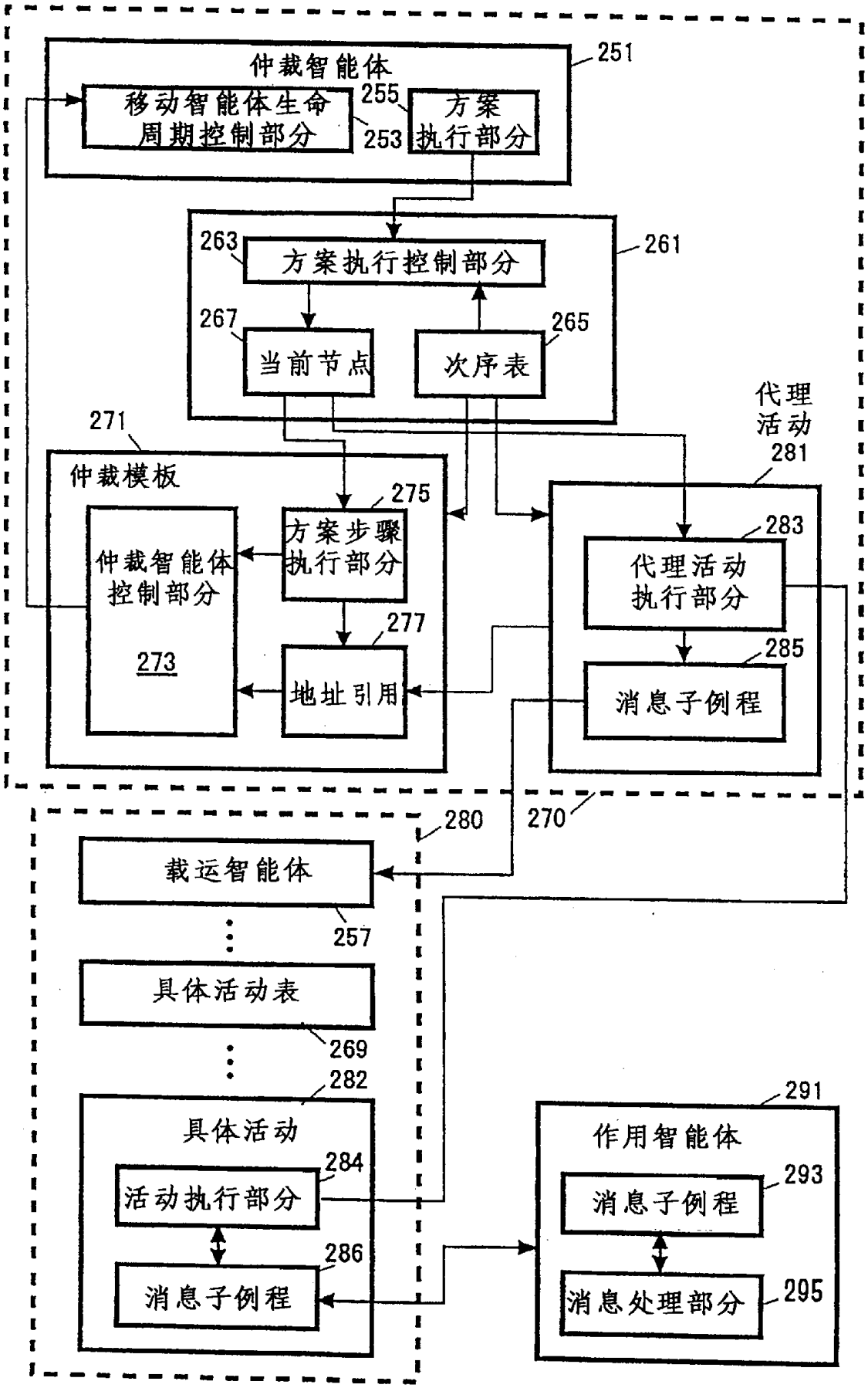

仲裁智能体
251
移动智能体生命周期控制部分
255
253
方案执行部分
263
方案执行控制部分
261
267
当前节点
265
次序表
代理活动
281
271
仲裁模板
275
仲裁智能体控制部分
273
方案步骤执行部分
283
代理活动执行部分
277
地址引用
285
消息子例程
270
280
载运智能体
257
具体活动表
269
282
具体活动
284
活动执行部分
286
消息子例程
291
作用智能体
293
消息子例程
消息处理部分
295

图 7

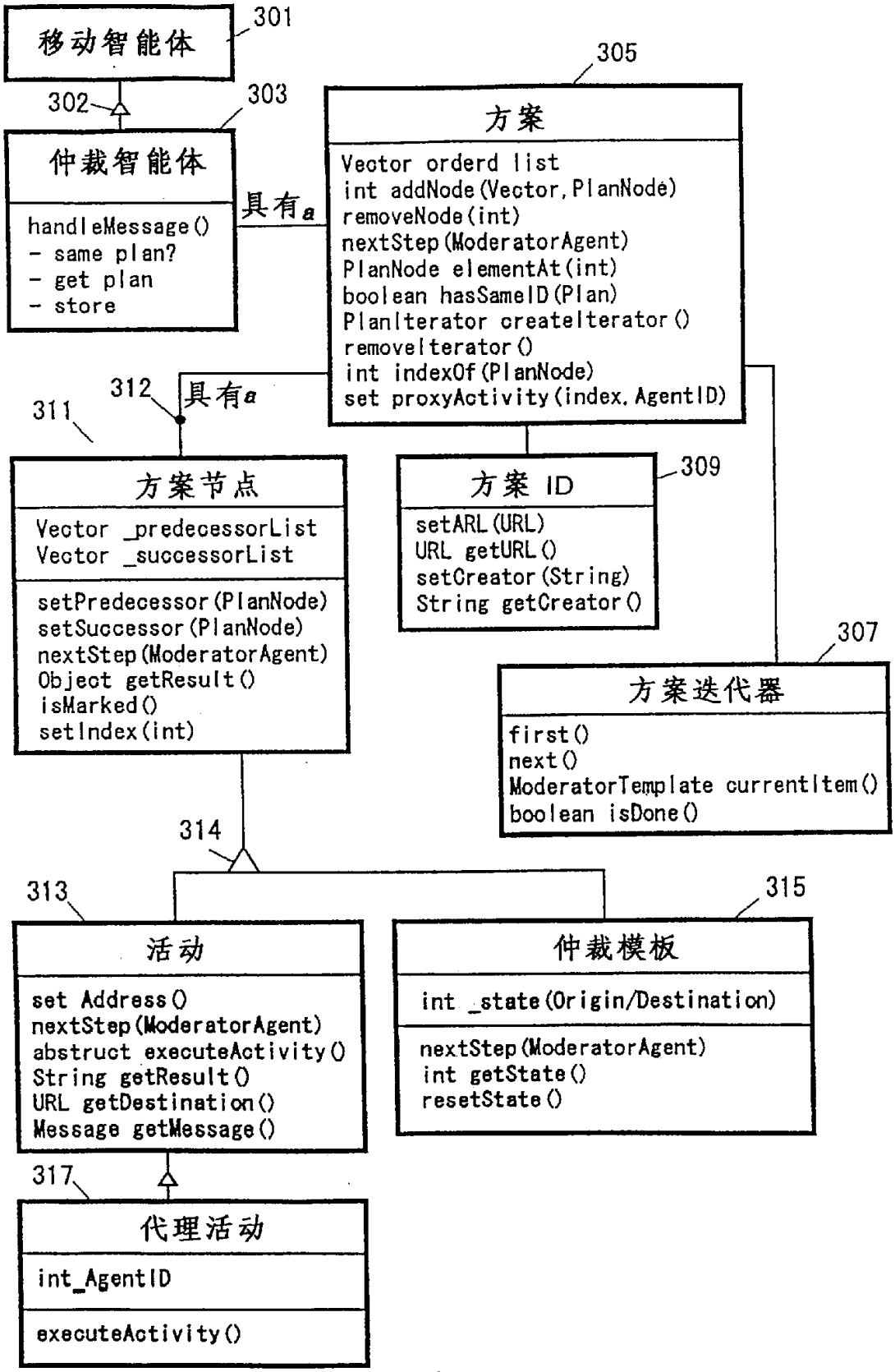

移动智能体
301
302
303
仲裁智能体
handleMessage()
- same plan?
- get plan
- store
具有a
305
方案
Vector orderd list
int addNode(Vector, PlanNode)
removeNode(int)
nextStep(ModeratorAgent)
PlanNode elementAt(int)
boolean hasSameID(Plan)
PlanIterator createIterator()
removeIterator()
int indexOf(PlanNode)
set proxyActivity(index, AgentID)
312
具有a
311
方案节点
Vector _predecessorList
Vector _successorList
setPredecessor(PlanNode)
setSuccessor(PlanNode)
nextStep(ModeratorAgent)
Object getResult()
isMarked()
setIndex(int)
方案 ID
309
setARL(URL)
URL getURL()
setCreator(String)
String getCreator()
307
方案迭代器
first()
next()
ModeratorTemplate currentItem()
boolean isDone()
314
313
活动
set Address()
nextStep(ModeratorAgent)
abstruct executeActivity()
String getResult()
URL getDestination()
Message getMessage()
315
仲裁模板
int _state(Origin/Destination)
nextStep(ModeratorAgent)
int getState()
resetState()
317
代理活动
int_AgentID
executeActivity()

图8

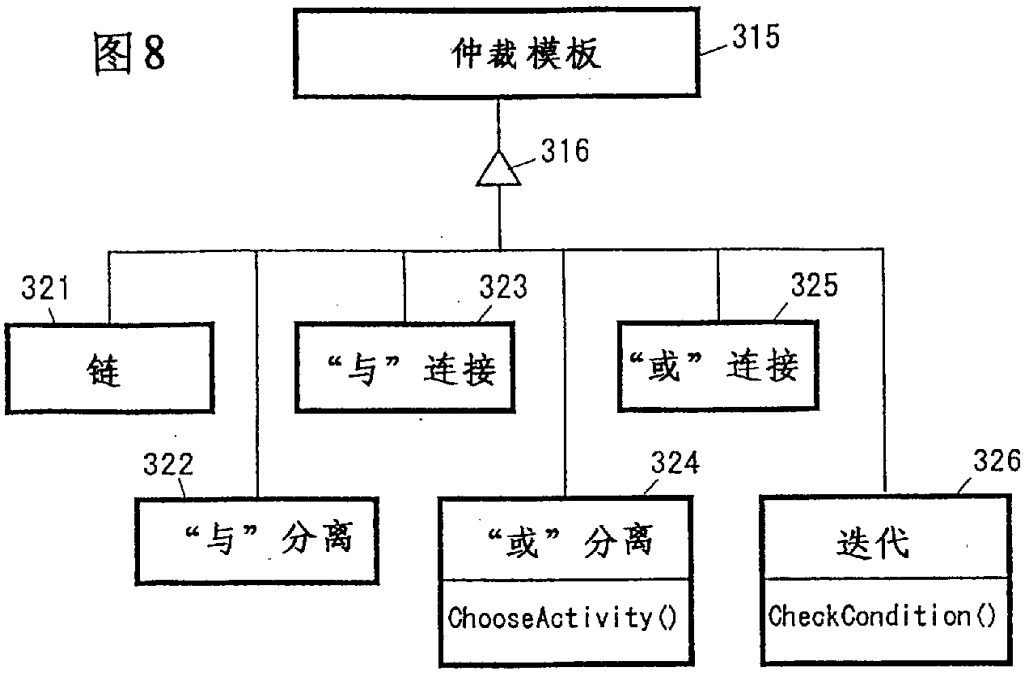

仲裁模板
315
316
321
链
323
“与”连接
325
“或”连接
322
“与”分离
324
“或”分离
ChooseActivity()
326
迭代
CheckCondition()

图10

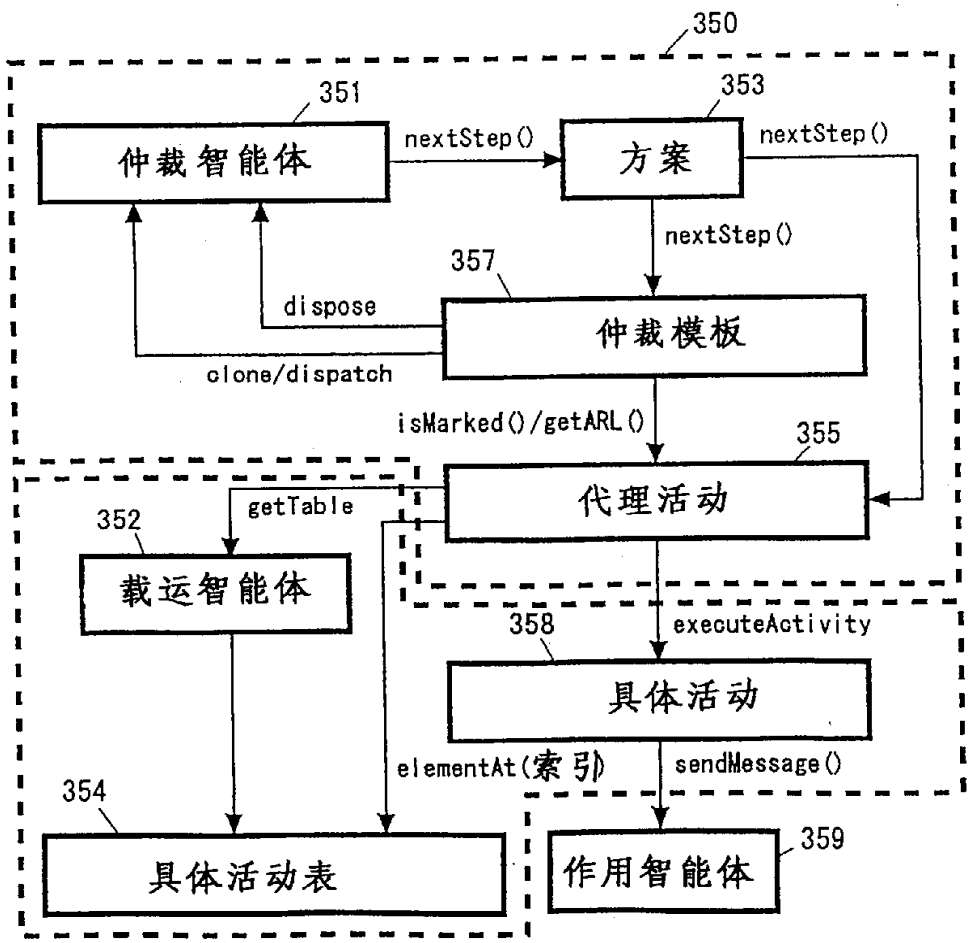

350
351
仲裁智能体
nextStep()
353
方案
nextStep()
nextStep()
357
dispose
仲裁模板
clone/dispatch
isMarked()/getARL()
355
代理活动
getTable
352
载运智能体
358
executeActivity
具体活动
elementAt(索引)
sendMessage()
354
具体活动表
359
作用智能体

图9

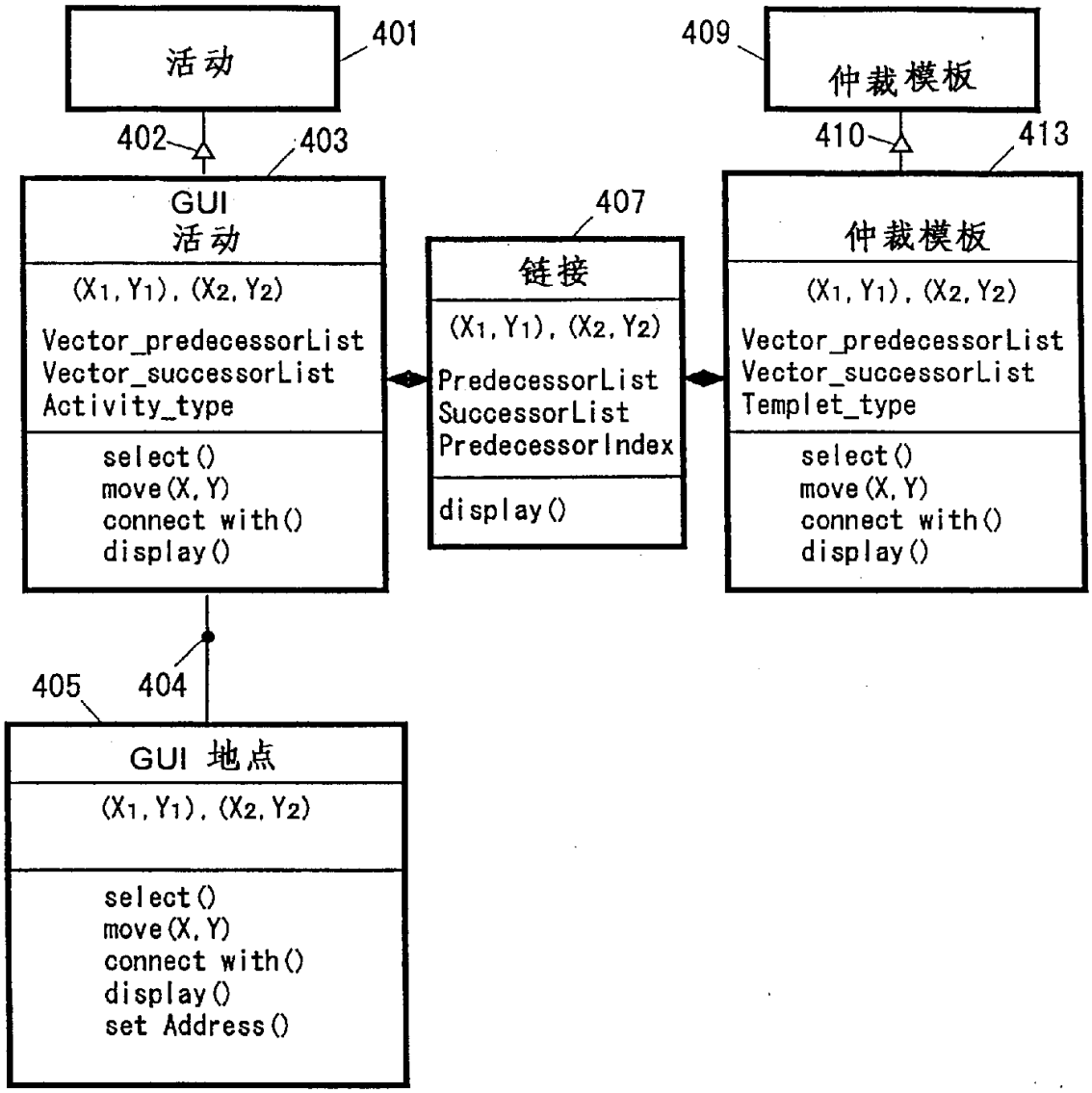

活动
401
409
仲裁模板
402
403
410
413
GUI
活动
(X1, Y1), (X2, Y2)
Vector_predecessorList
Vector_successorList
Activity_type
select()
move(X, Y)
connect with()
display()
407
链接
(X1, Y1), (X2, Y2)
PredecessorList
SuccessorList
PredecessorIndex
display()
仲裁模板
(X1, Y1), (X2, Y2)
Vector_predecessorList
Vector_successorList
Templet_type
select()
move(X, Y)
connect with()
display()
405
404
GUI 地点
(X1, Y1), (X2, Y2)
select()
move(X, Y)
connect with()
display()
set Address()

图11

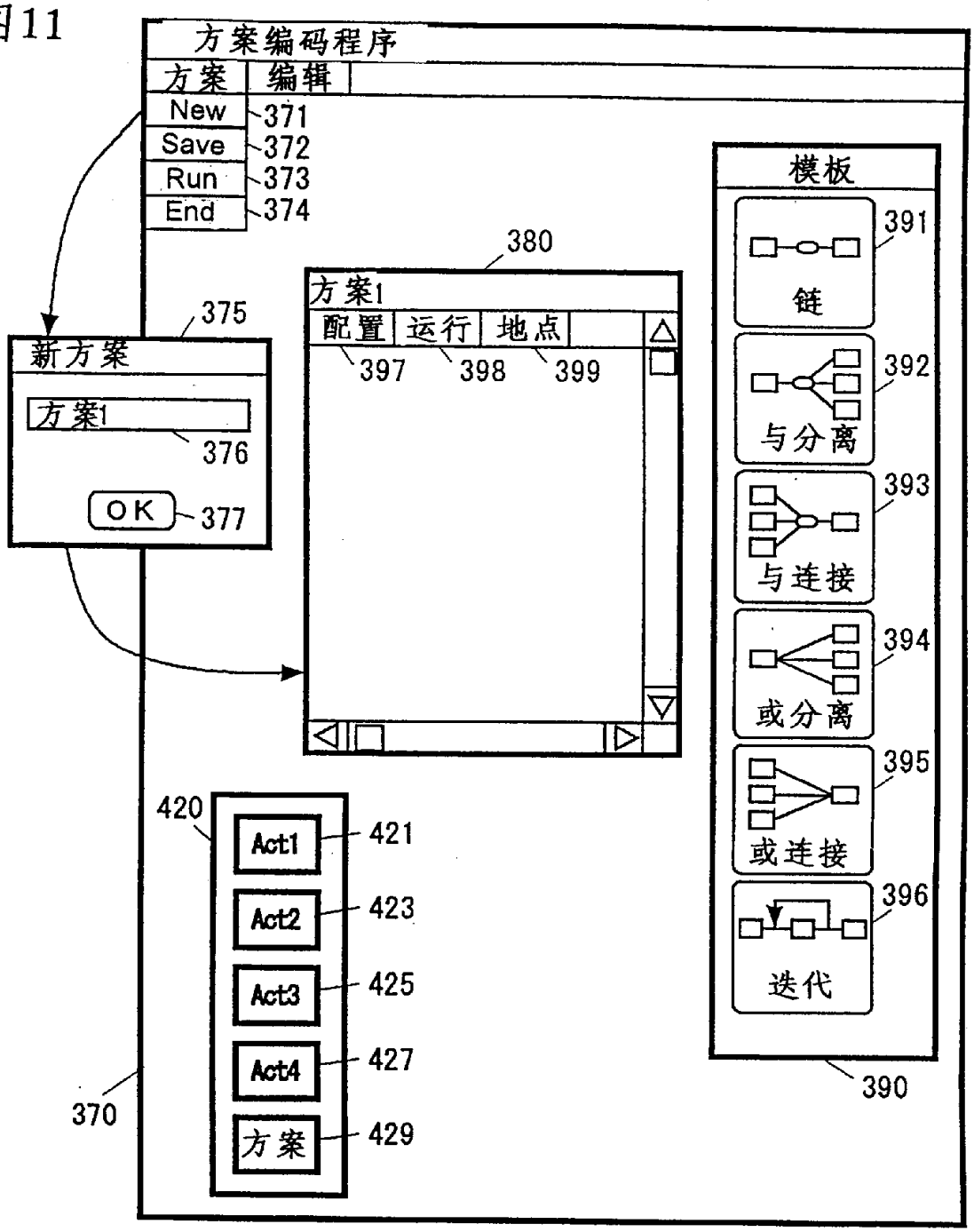

方案编码程序
方案
编辑
New
371
Save
372
Run
373
End
374
380
方案1
配置
运行
地点
397
398
399
375
新方案
方案1
376
OK
377
模板
链
391
与分离
392
与连接
393
或分离
394
或连接
395
迭代
396
390
420
Act1
421
Act2
423
Act3
425
Act4
427
方案
429
370

图12

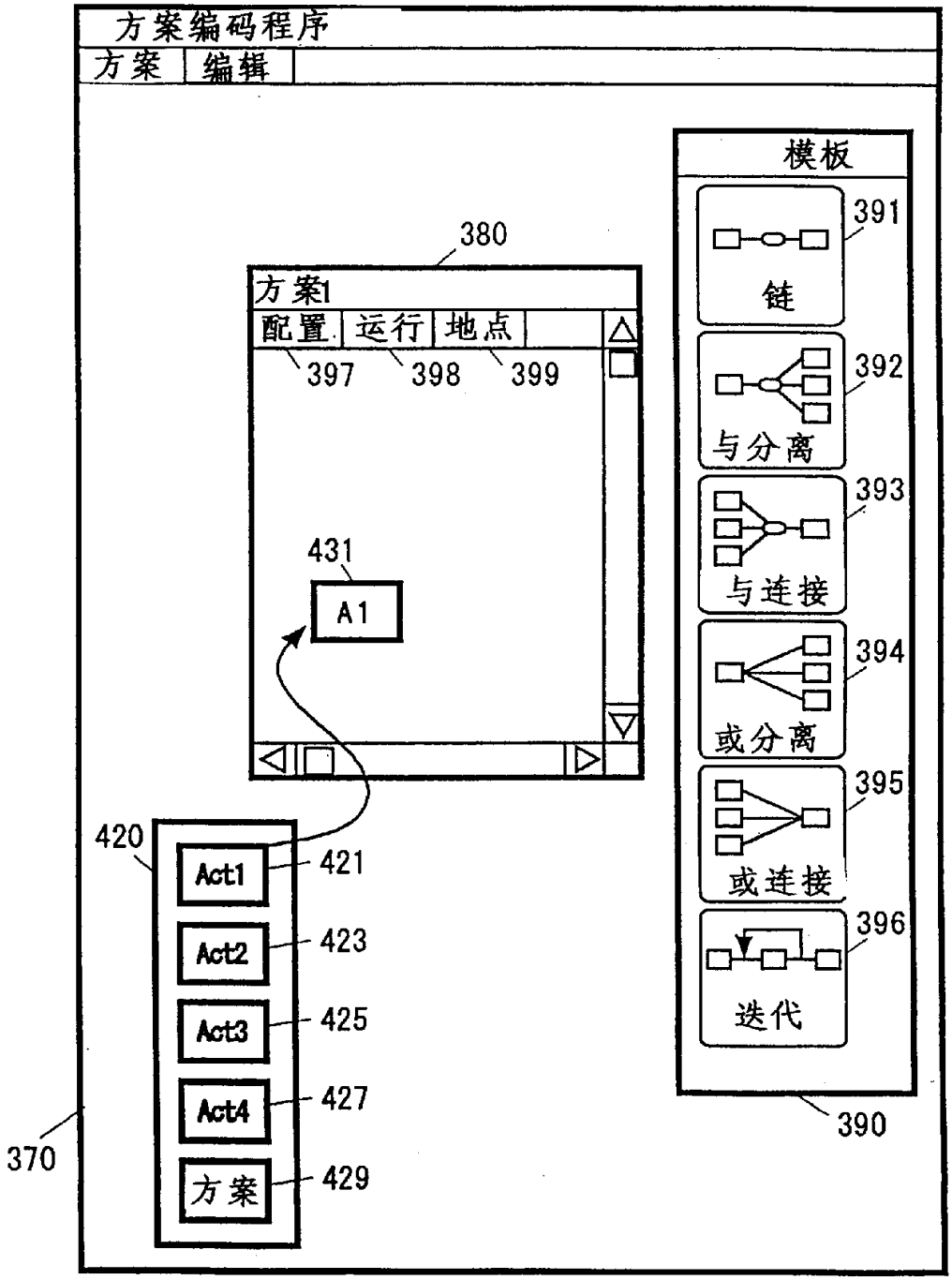

方案编码程序
方案
编辑
模板
391
链
392
与分离
393
与连接
394
或分离
395
或连接
396
迭代
390
380
方案1
配置
运行
地点
397
398
399
431
A1
420
Act1
421
Act2
423
Act3
425
Act4
427
方案
429
370

图13

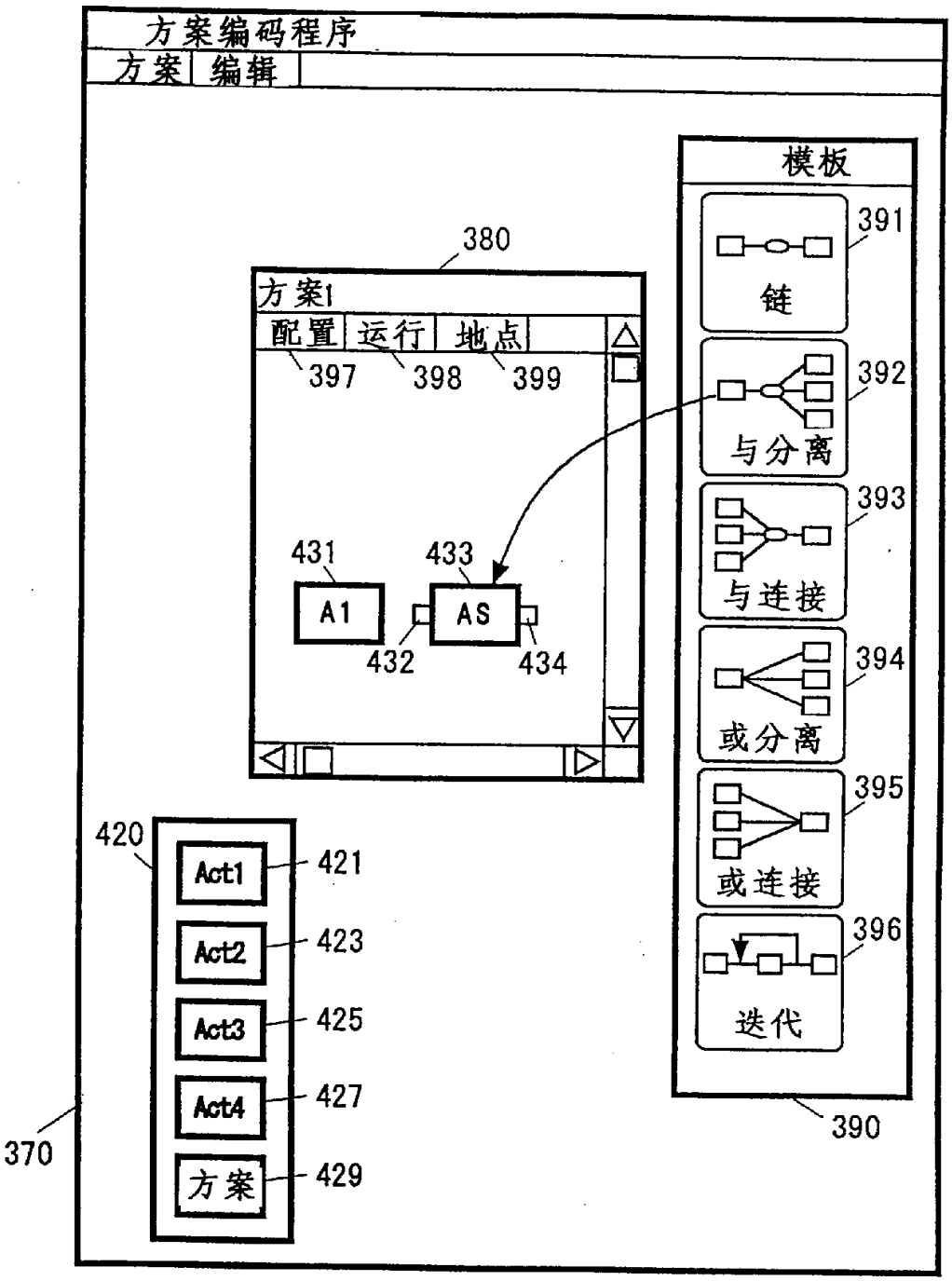

方案编码程序
方案
编辑
模板
391
链
392
与分离
393
与连接
394
或分离
395
或连接
396
迭代
390
380
方案1
配置
运行
地点
397
398
399
431
A1
433
AS
432
434
420
Act1
421
Act2
423
Act3
425
Act4
427
方案
429
370

图14

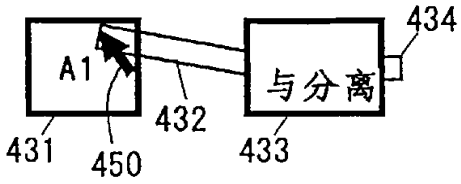

A1
与分离
434
431
450
432
433

图15

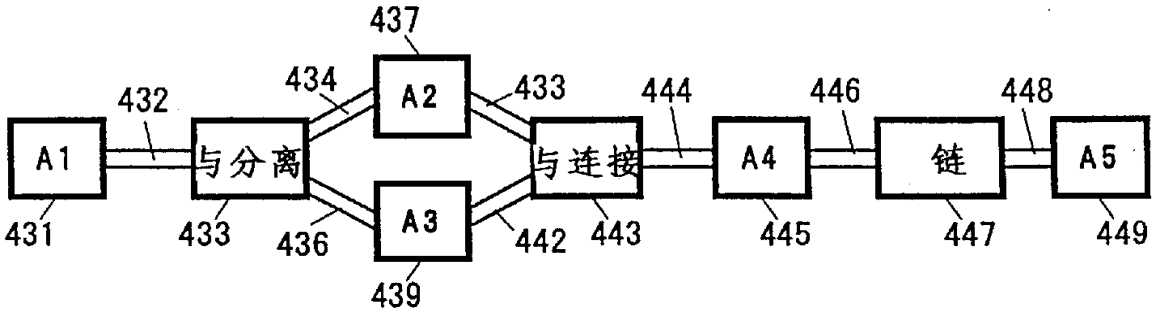

437
434
A2
433
444
446
448
432
A1
与分离
与连接
A4
链
A5
431
433
436
A3
442
443
445
447
449
439

图16

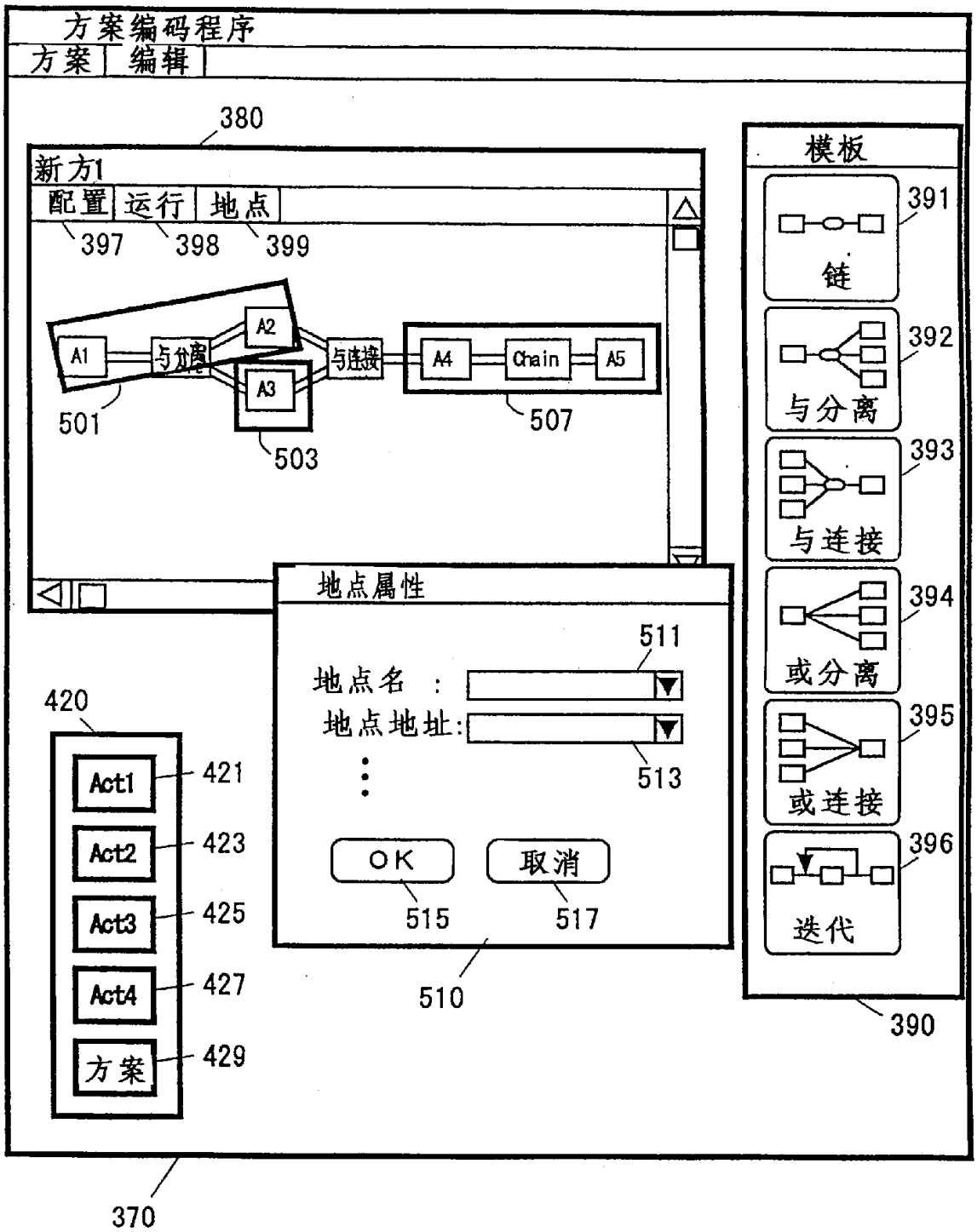

方案编码程序
方案
编辑
380
新方1
配置
运行
地点
397
398
399
A1
与分离
A2
A3
与连接
A4
Chain
A5
501
503
507
模板
391
链
392
与分离
393
与连接
394
或分离
395
或连接
396
迭代
390
地点属性
511
地点名 :
地点地址:
513
OK
取消
515
517
510
420
Act1
421
Act2
423
Act3
425
Act4
427
方案
429
370

图17

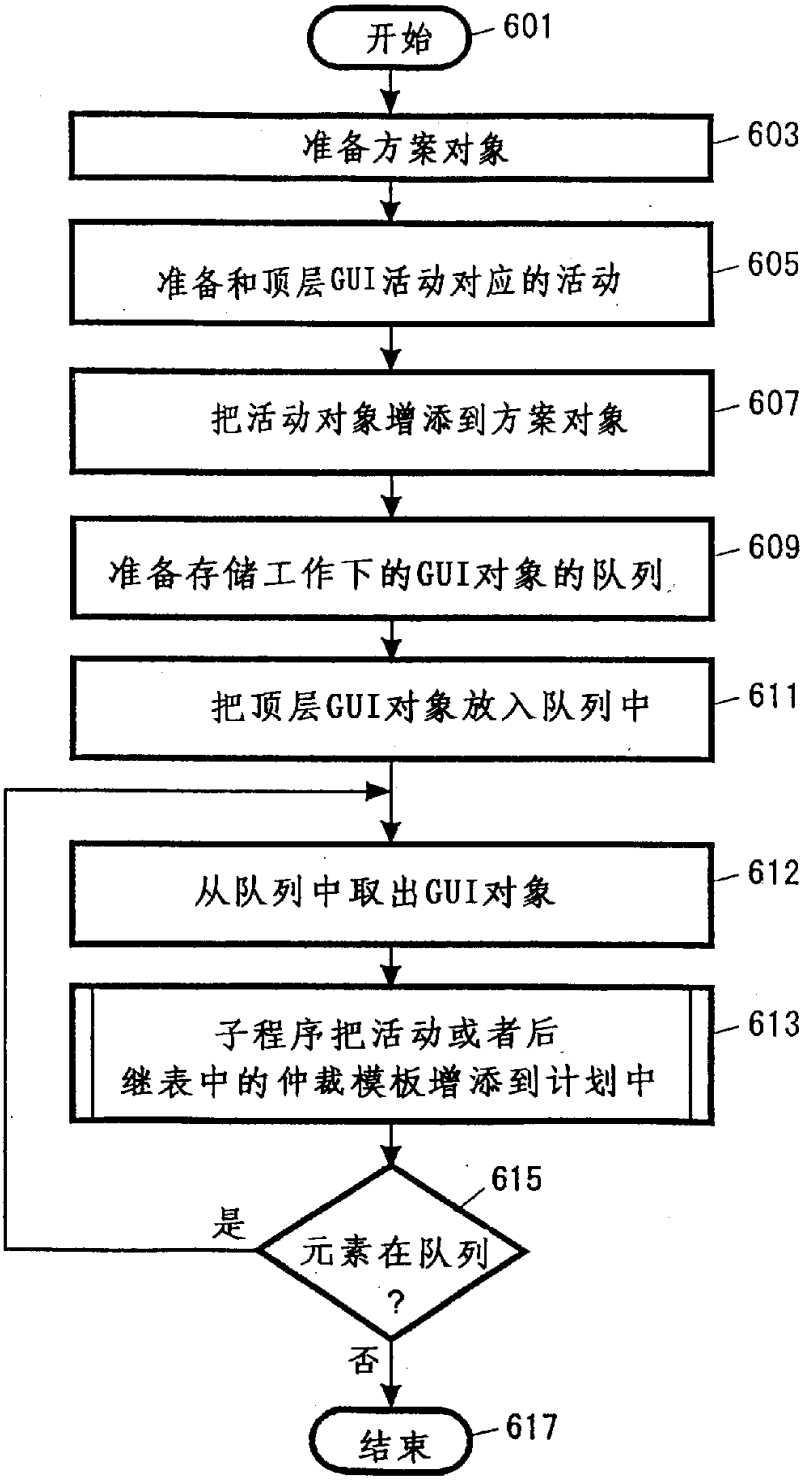

开始
601
准备方案对象
603
准备和顶层GUI活动对应的活动
605
把活动对象增添到方案对象
607
准备存储工作下的GUI对象的队列
609
把顶层GUI对象放入队列中
611
从队列中取出GUI对象
612
子程序把活动或者后
继表中的仲裁模板增添到计划中
613
615
是
元素在队列
?
否
结束
617

图18

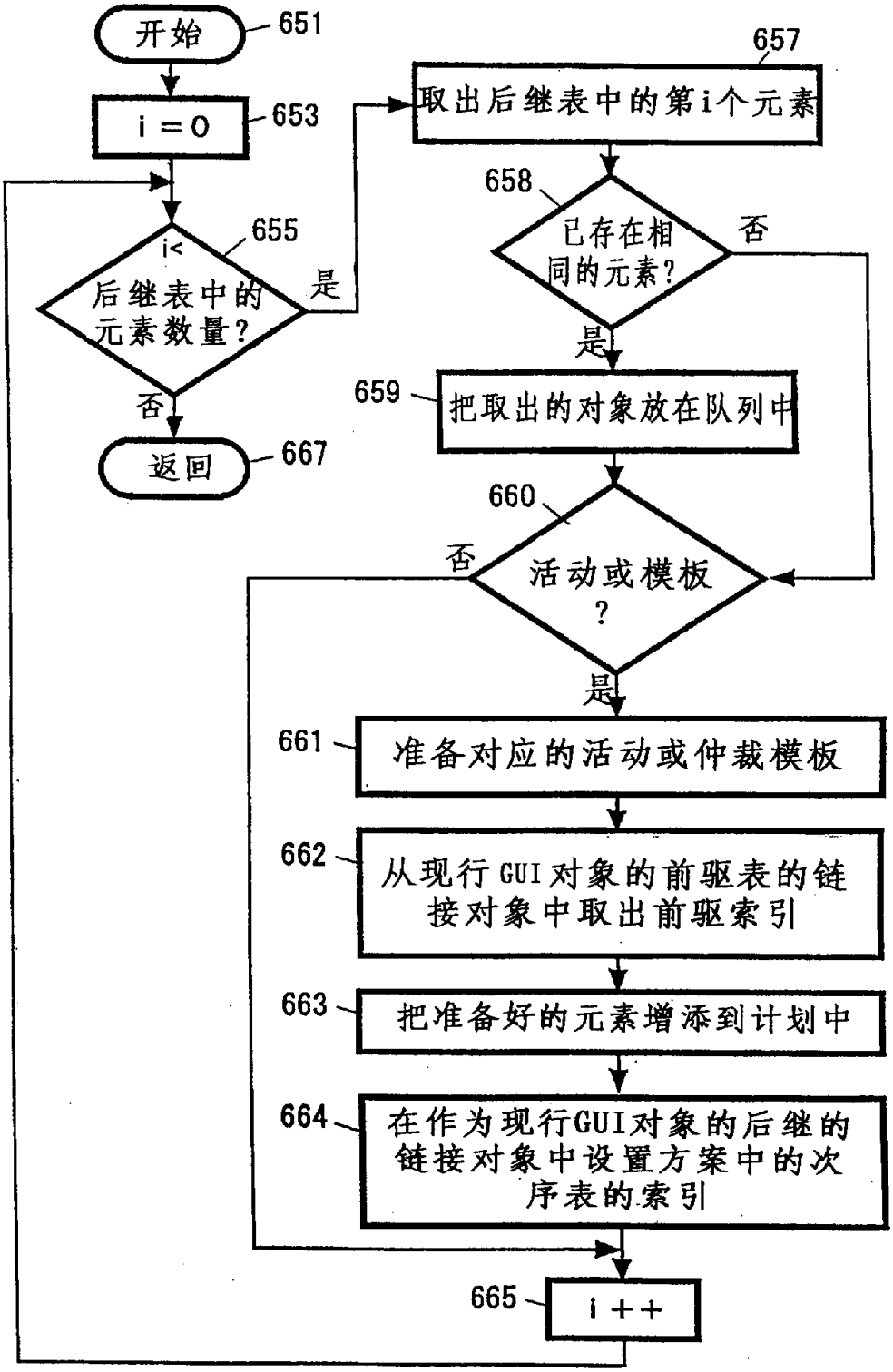

开始
651
i = 0
653
i<
后继表中的
元素数量?
655
是
否
返回
667
657
取出后继表中的第i个元素
658
已存在相
同的元素?
否
是
659
把取出的对象放在队列中
660
否
活动或模板
?
是
661
准备对应的活动或仲裁模板
662
从现行GUI对象的前驱表的链
接对象中取出前驱索引
663
把准备好的元素增添到计划中
664
在作为现行GUI对象的后继的
链接对象中设置方案中的次
序表的索引
665
i++

图19

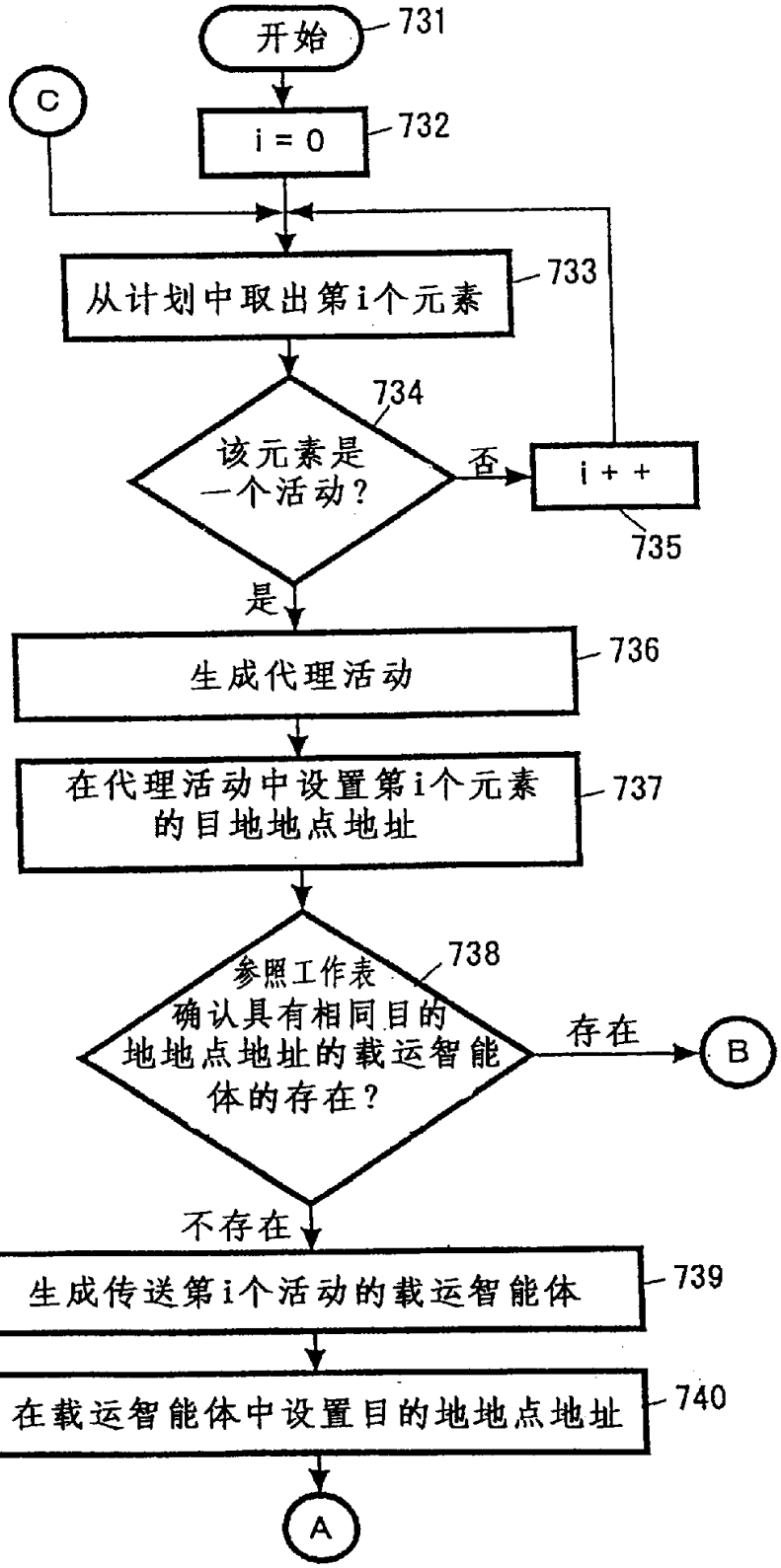

开始
731
C
i = 0
732
从计划中取出第i个元素
733
734
该元素是一个活动？
否
i + +
735
是
生成代理活动
736
在代理活动中设置第i个元素的目地地点地址
737
参照工作表确认具有相同目的地地点地址的载运智能体的存在？
738
存在
B
不存在
生成传送第i个活动的载运智能体
739
在载运智能体中设置目的地地点地址
740
A

图20

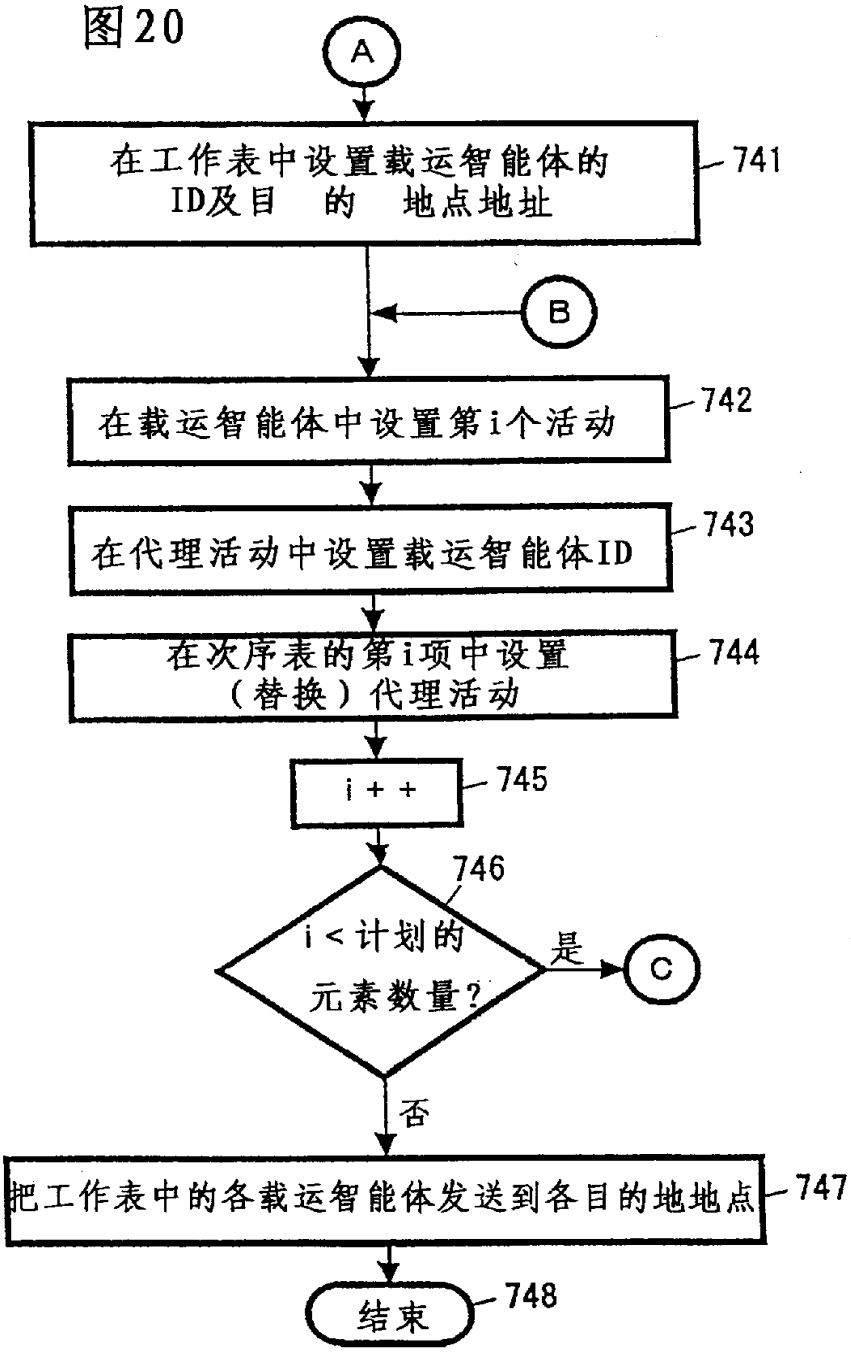

A
在工作表中设置载运智能体的ID及目 的 地点地址
741
B
在载运智能体中设置第i个活动
742
在代理活动中设置载运智能体ID
743
在次序表的第i项中设置（替换）代理活动
744
i + +
745
746
i < 计划的元素数量?
是
C
否
把工作表中的各载运智能体发送到各目的地地点
747
结束
748

图21

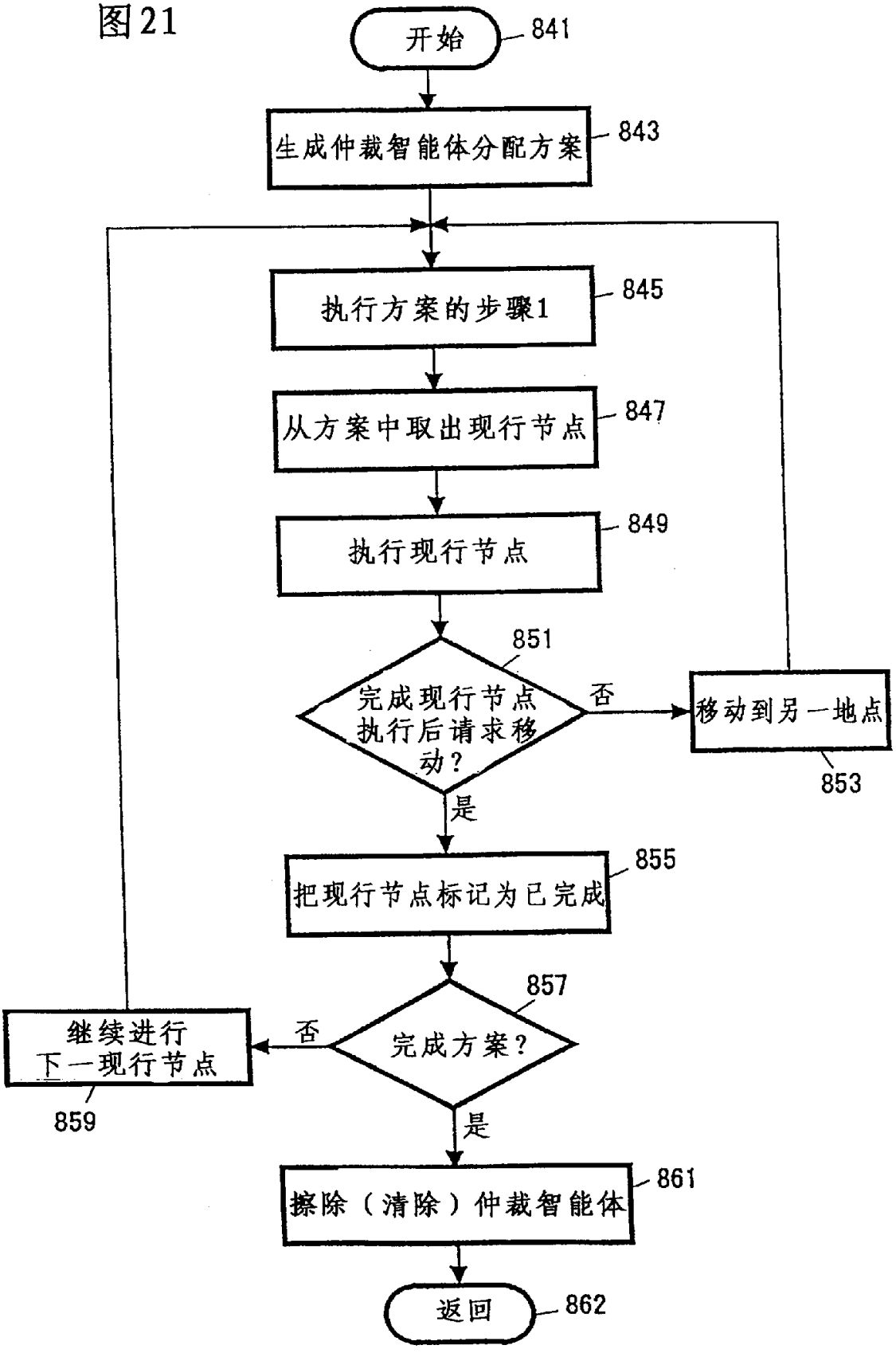

开始
841
生成仲裁智能体分配方案
843
执行方案的步骤1
845
从方案中取出现行节点
847
执行现行节点
849
851
完成现行节点执行后请求移动？
否
移动到另一地点
853
是
把现行节点标记为已完成
855
857
完成方案？
否
继续进行下一现行节点
859
是
擦除（清除）仲裁智能体
861
返回
862

图22

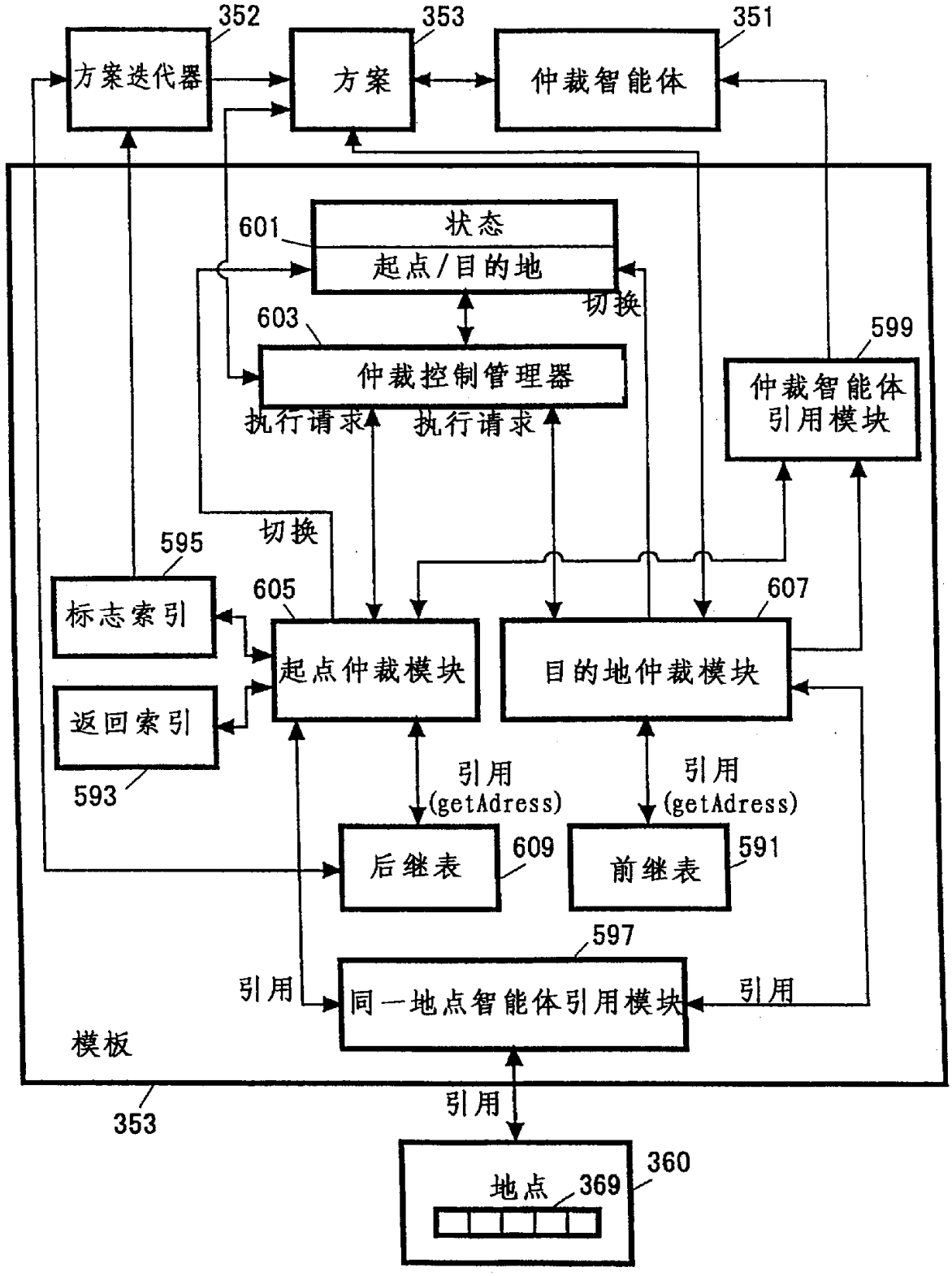

352
方案迭代器
353
方案
351
仲裁智能体
601
状态
起点/目的地
切换
603
仲裁控制管理器
执行请求
执行请求
599
仲裁智能体
引用模块
切换
595
标志索引
605
起点仲裁模块
607
目的地仲裁模块
返回索引
593
引用
(getAdress)
引用
(getAdress)
后继表
609
前继表
591
597
引用
同一地点智能体引用模块
引用
模板
353
引用
360
地点
369

图23

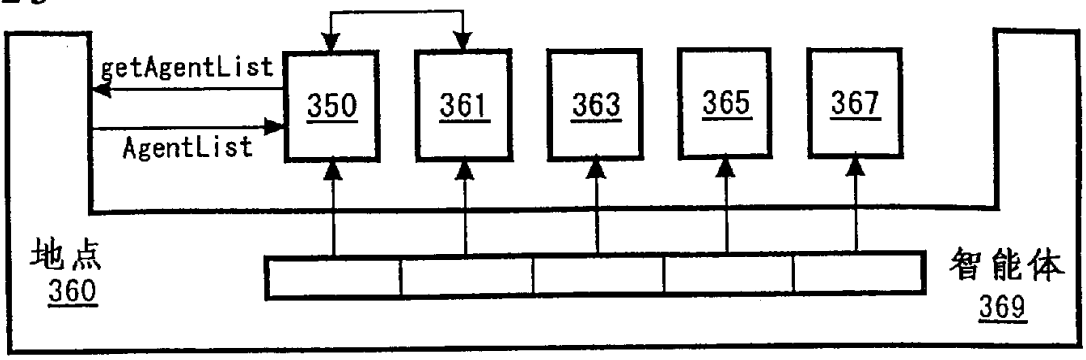

getAgentList
AgentList
350
361
363
365
367
地点
360
智能体
369

图34

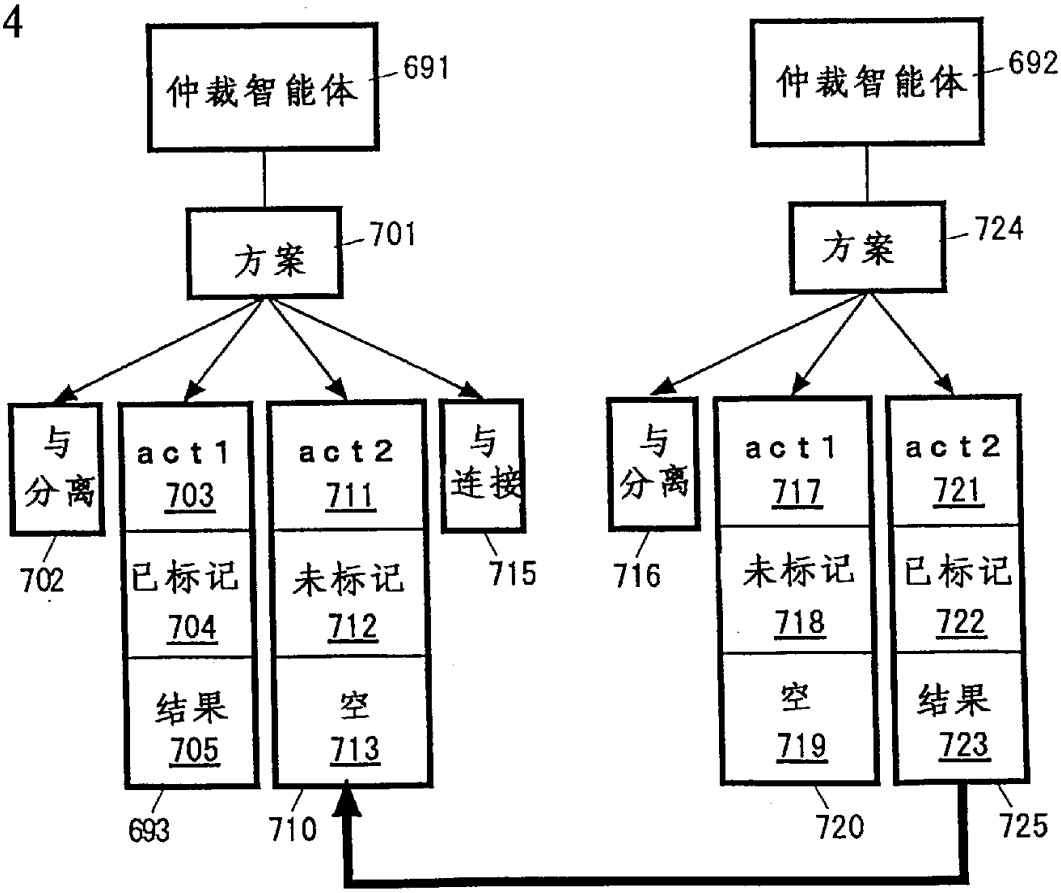

仲裁智能体
691
方案
701
与
分离
act1
703
act2
711
与
连接
已标记
704
未标记
712
结果
705
空
713
702
715
693
710
仲裁智能体
692
方案
724
与
分离
act1
717
act2
721
未标记
718
已标记
722
空
719
结果
723
716
720
725

图24

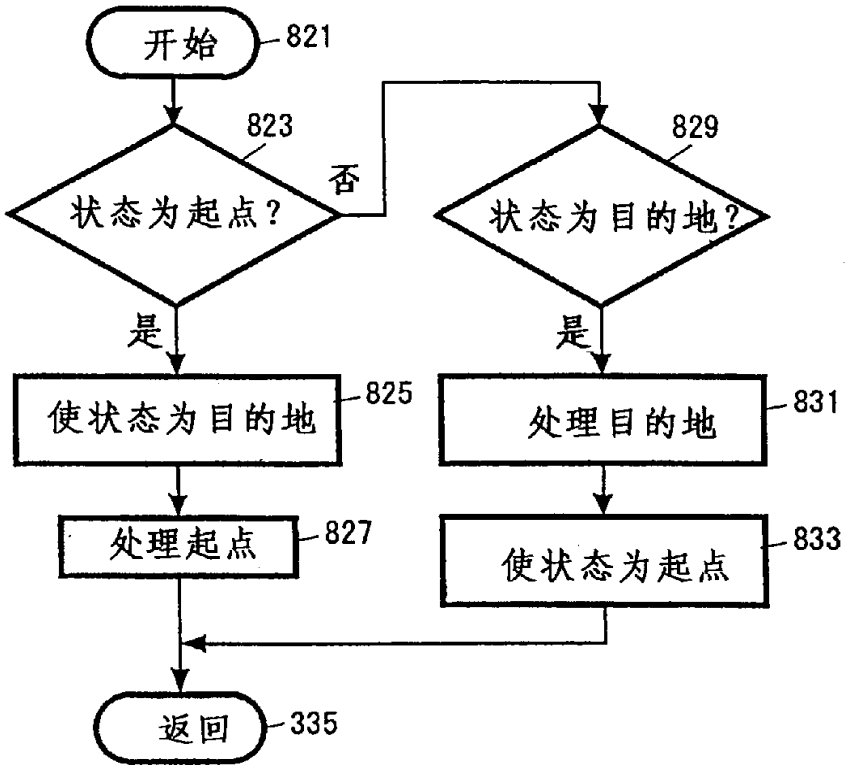

开始
821
823
状态为起点？
否
829
状态为目的地？
是
是
使状态为目的地
825
处理目的地
831
处理起点
827
使状态为起点
833
返回
335

图27

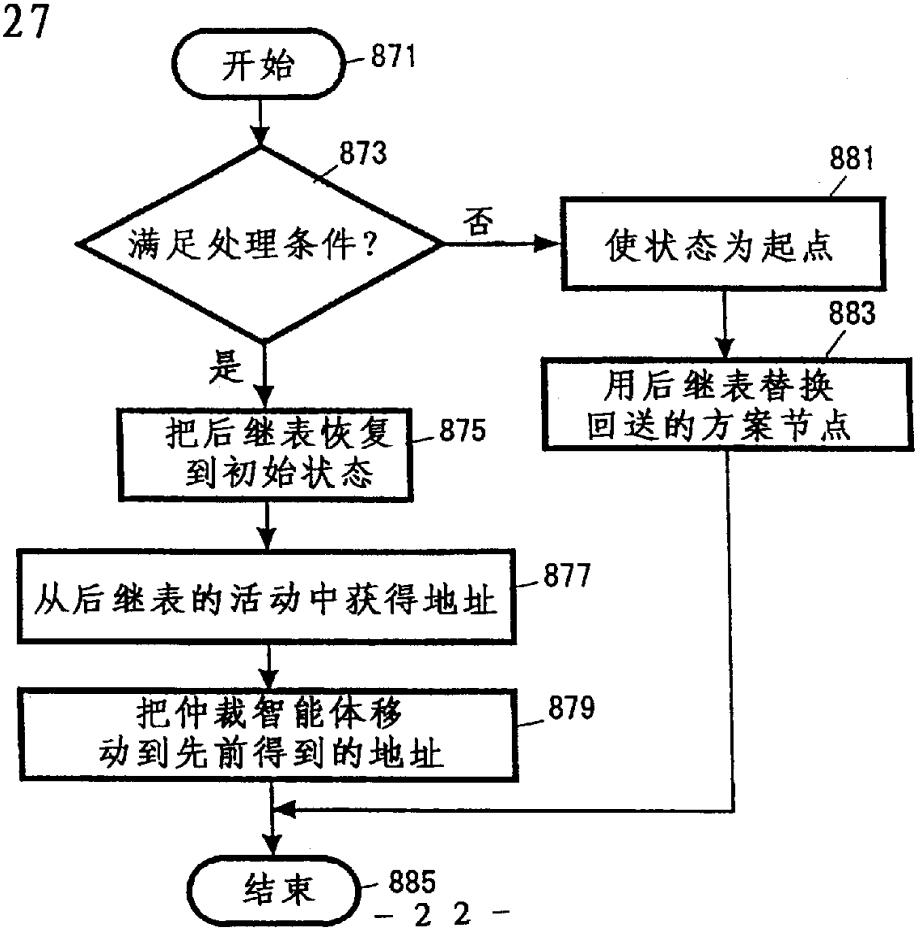

开始
871
873
满足处理条件？
否
881
使状态为起点
883
用后继表替换
回送的方案节点
是
把后继表恢复
到初始状态
875
从后继表的活动中获得地址
877
把仲裁智能体移
动到先前得到的地址
879
结束
885

图25

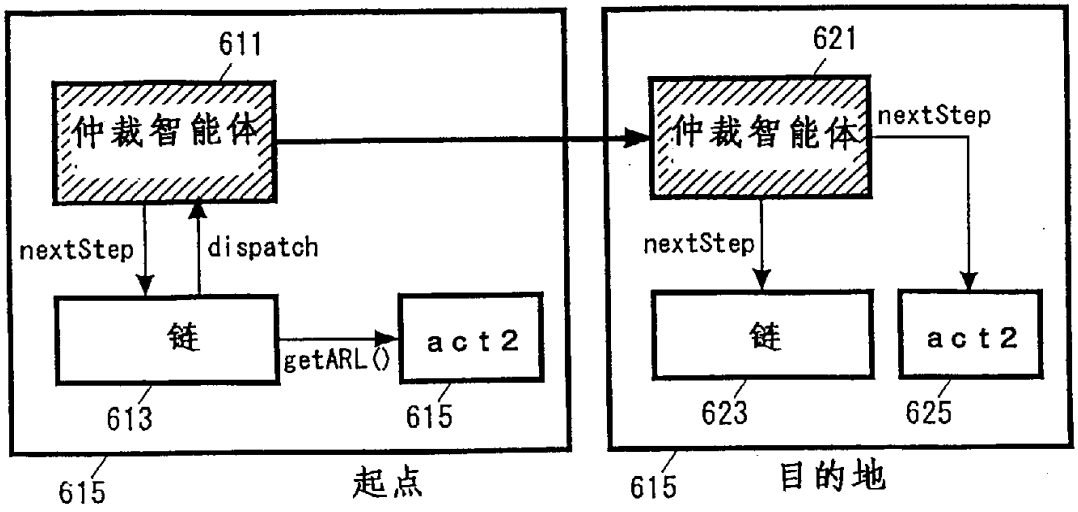

611
仲裁智能体
nextStep
dispatch
链
getARL()
act2
613
615
615
起点
621
仲裁智能体
nextStep
nextStep
链
act2
623
625
615
目的地

图26

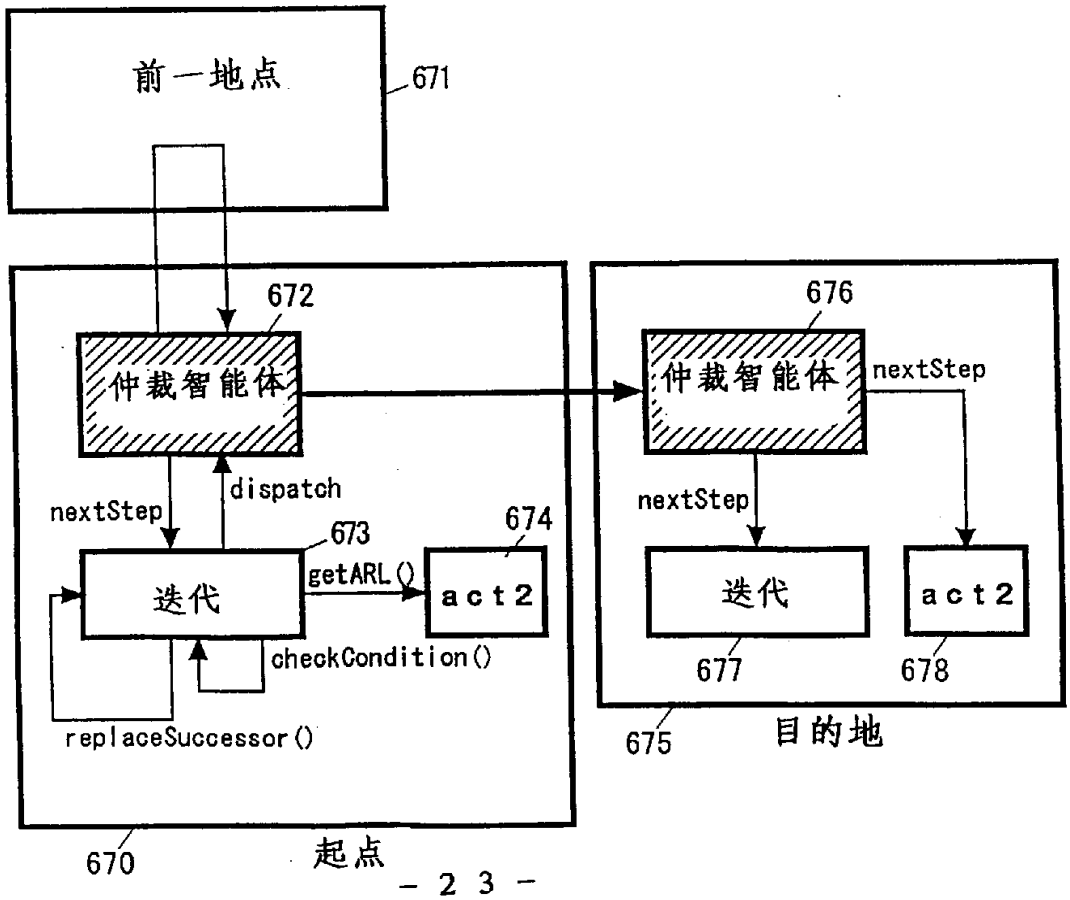

前一地点
671
672
仲裁智能体
nextStep
dispatch
673
迭代
getARL()
674
act2
checkCondition()
replaceSuccessor()
670
起点
676
仲裁智能体
nextStep
nextStep
迭代
act2
677
678
675
目的地

图28

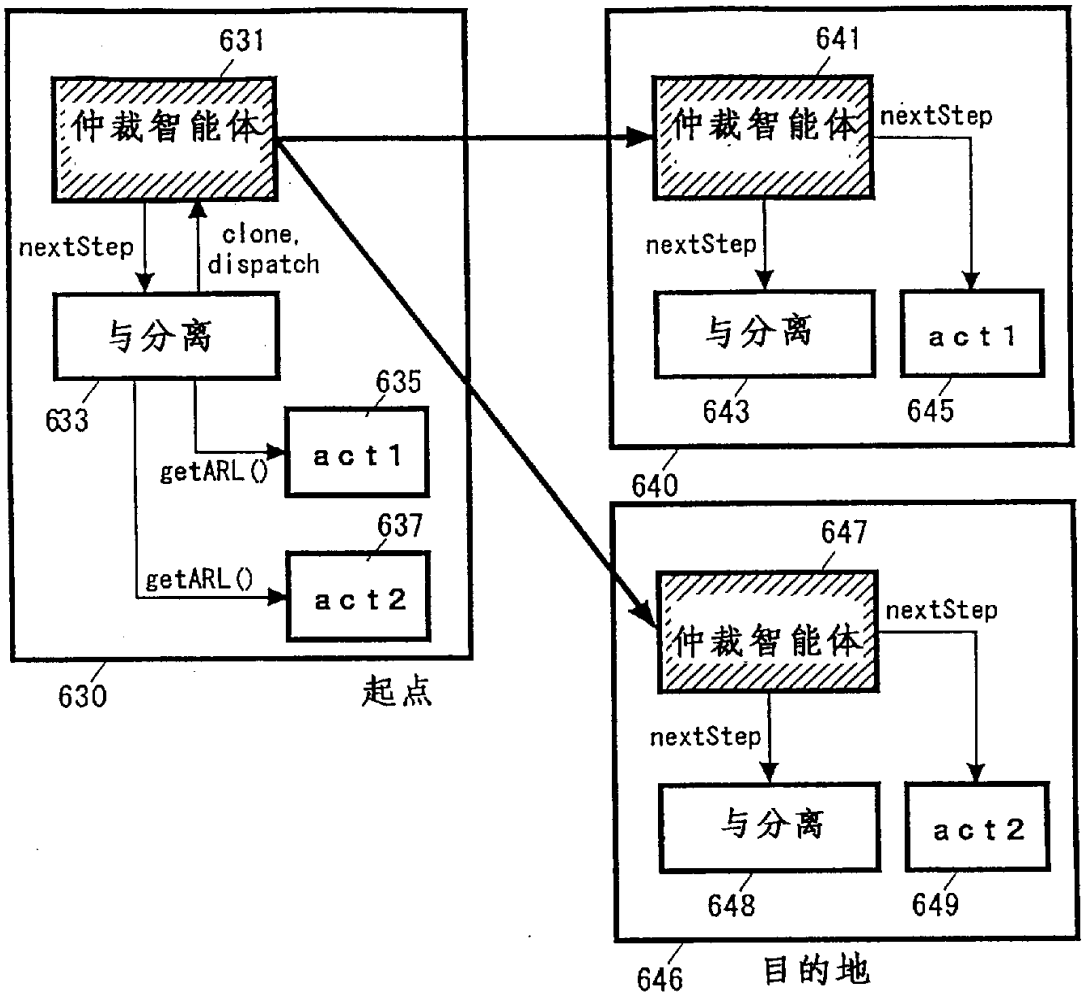

631
仲裁智能体
nextStep
clone,
dispatch
与分离
633
635
act1
getARL()
637
getARL()
act2
630
起点
641
仲裁智能体
nextStep
nextStep
与分离
act1
643
645
640
647
仲裁智能体
nextStep
nextStep
与分离
act2
648
649
646
目的地

图29

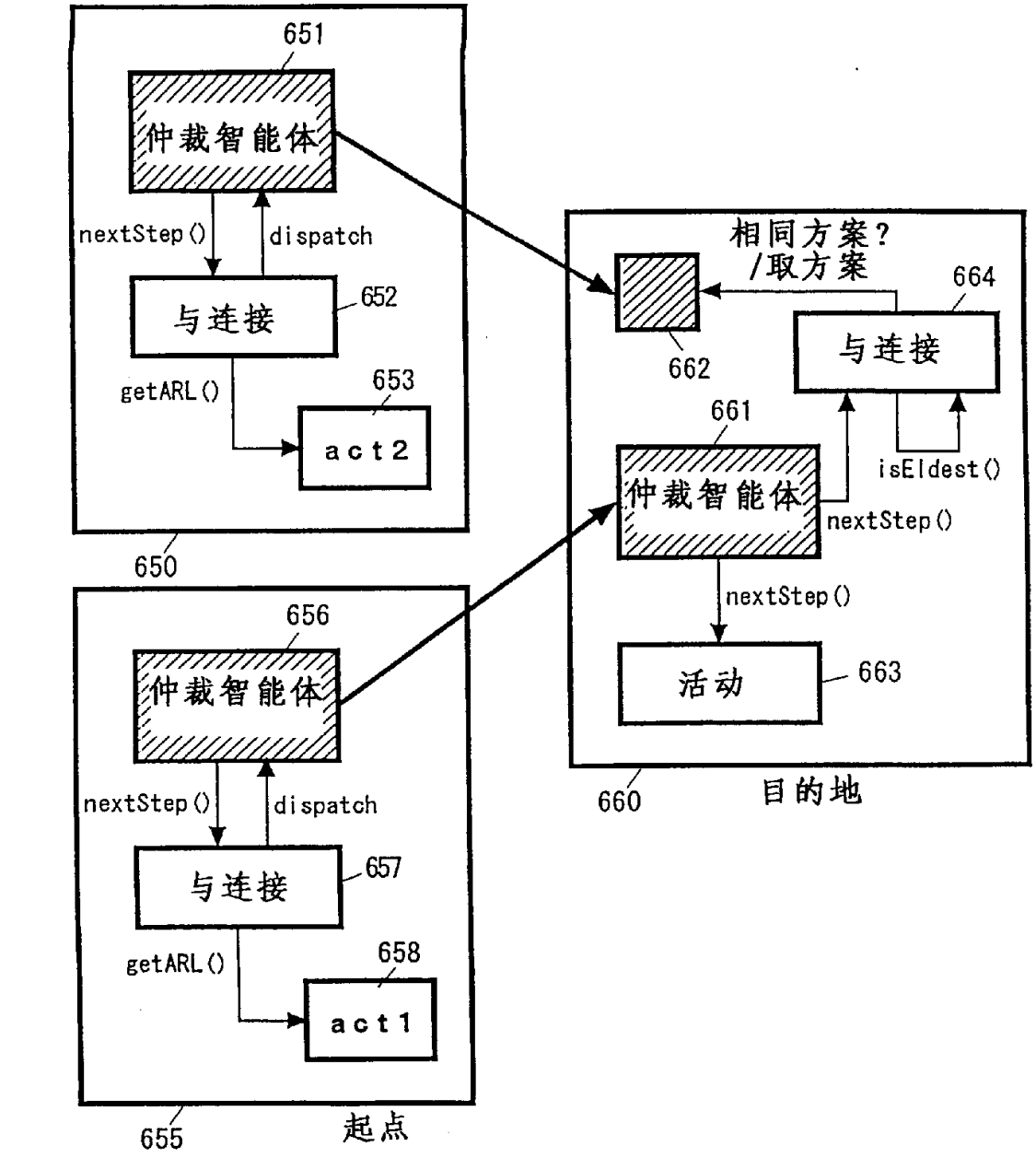

651
仲裁智能体
nextStep()
dispatch
与连接
652
getARL()
653
act2
650
656
仲裁智能体
nextStep()
dispatch
与连接
657
getARL()
658
act1
655
起点
相同方案?
/取方案
664
662
与连接
661
仲裁智能体
isEldest()
nextStep()
nextStep()
活动
663
660
目的地

图30

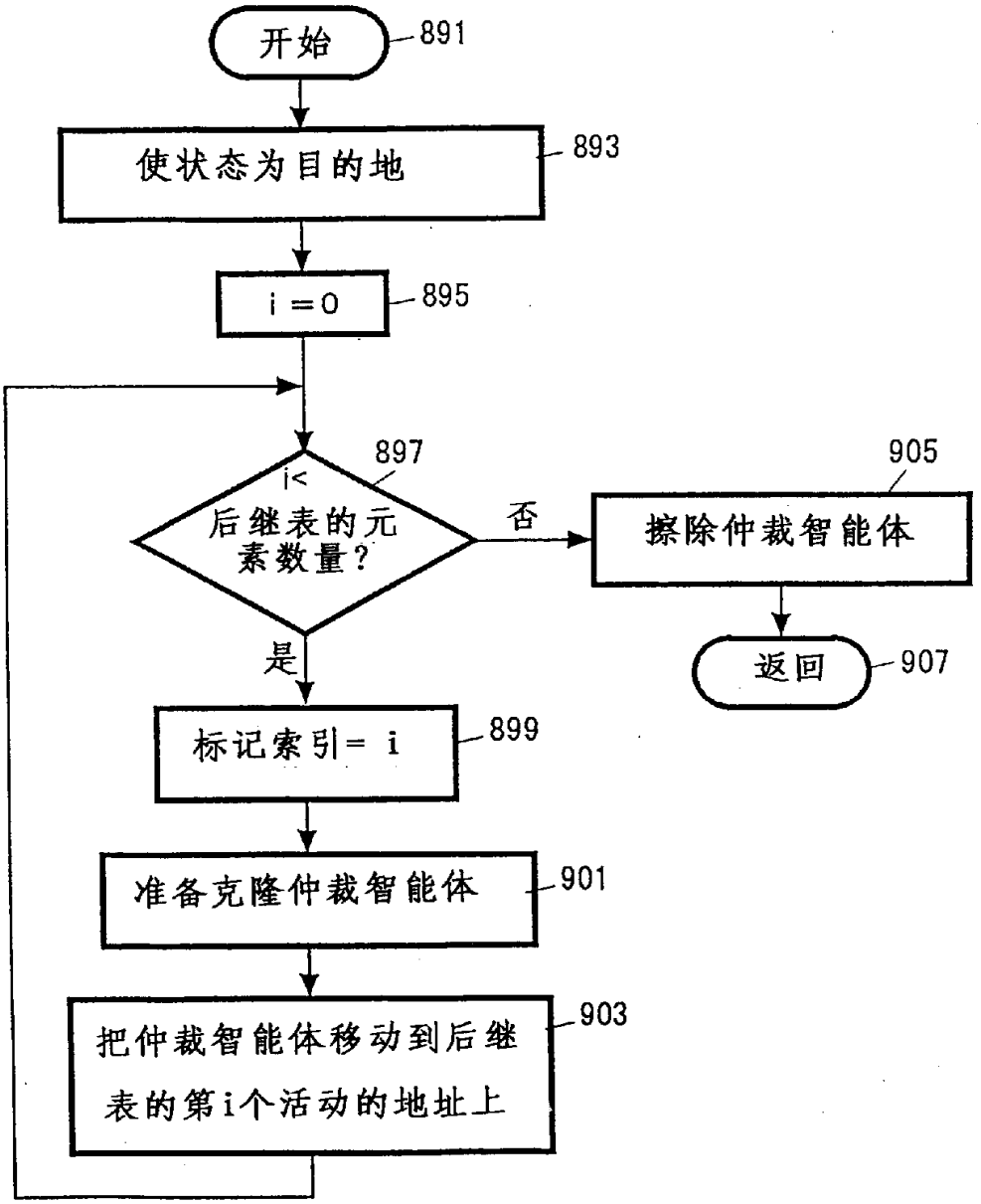

开始
891
使状态为目的地
893
i =0
895
i<
后继表的元素数量?
897
否
擦除仲裁智能体
905
返回
907
是
标记索引= i
899
准备克隆仲裁智能体
901
把仲裁智能体移动到后继表的第i个活动的地址上
903

图31

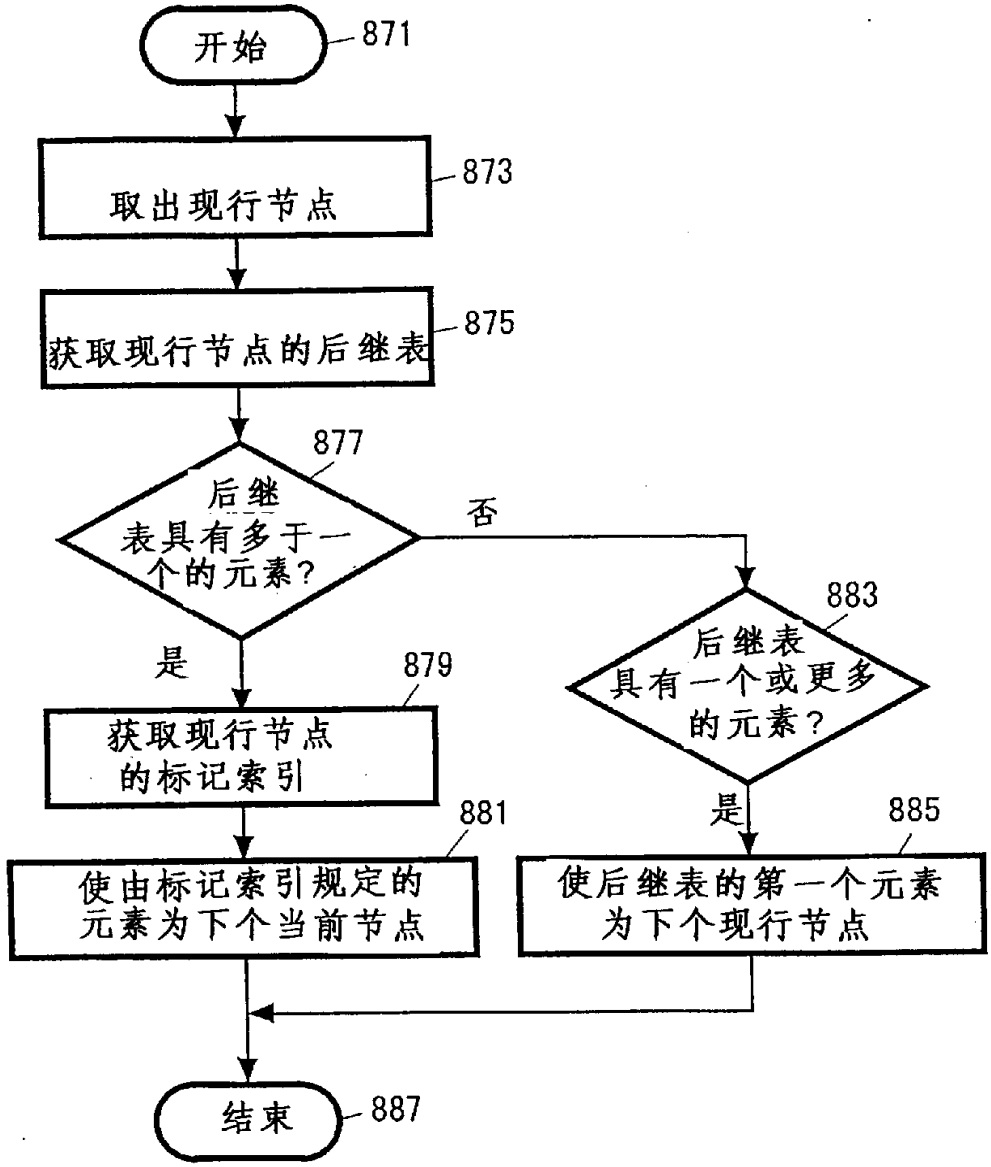

开始
871
取出现行节点
873
获取现行节点的后继表
875
877
后继表具有多于一个的元素?
否
是
883
后继表具有一个或更多的元素?
879
获取现行节点的标记索引
是
881
885
使由标记索引规定的元素为下个当前节点
使后继表的第一个元素为下个现行节点
结束
887

图 32

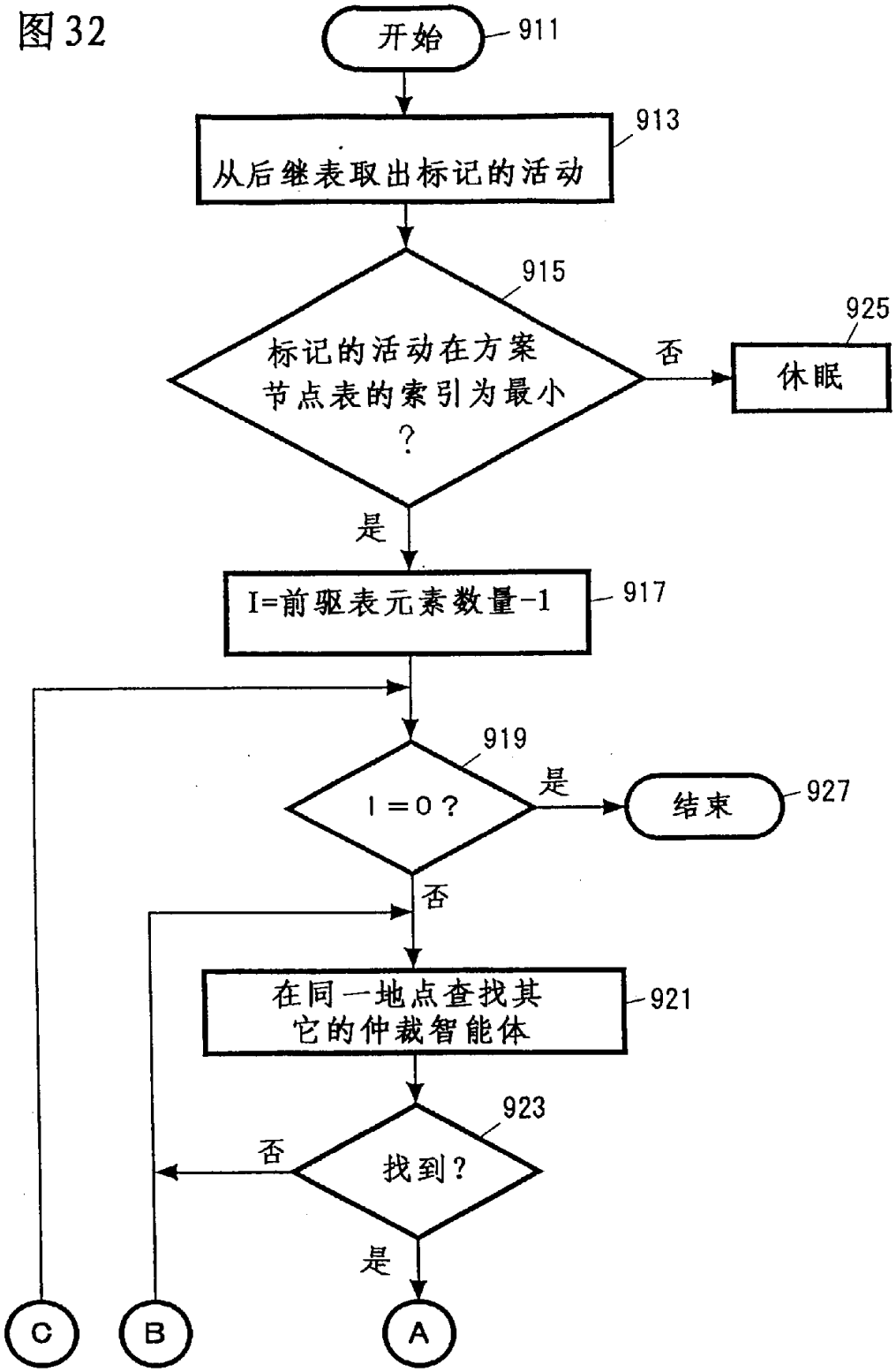

开始
911
从后继表取出标记的活动
913
标记的活动在方案节点表的索引为最小？
915
否
925
休眠
是
I=前驱表元素数量-1
917
I＝0？
919
是
结束
927
否
在同一地点查找其它的仲裁智能体
921
找到？
923
否
是
C
B
A

图33

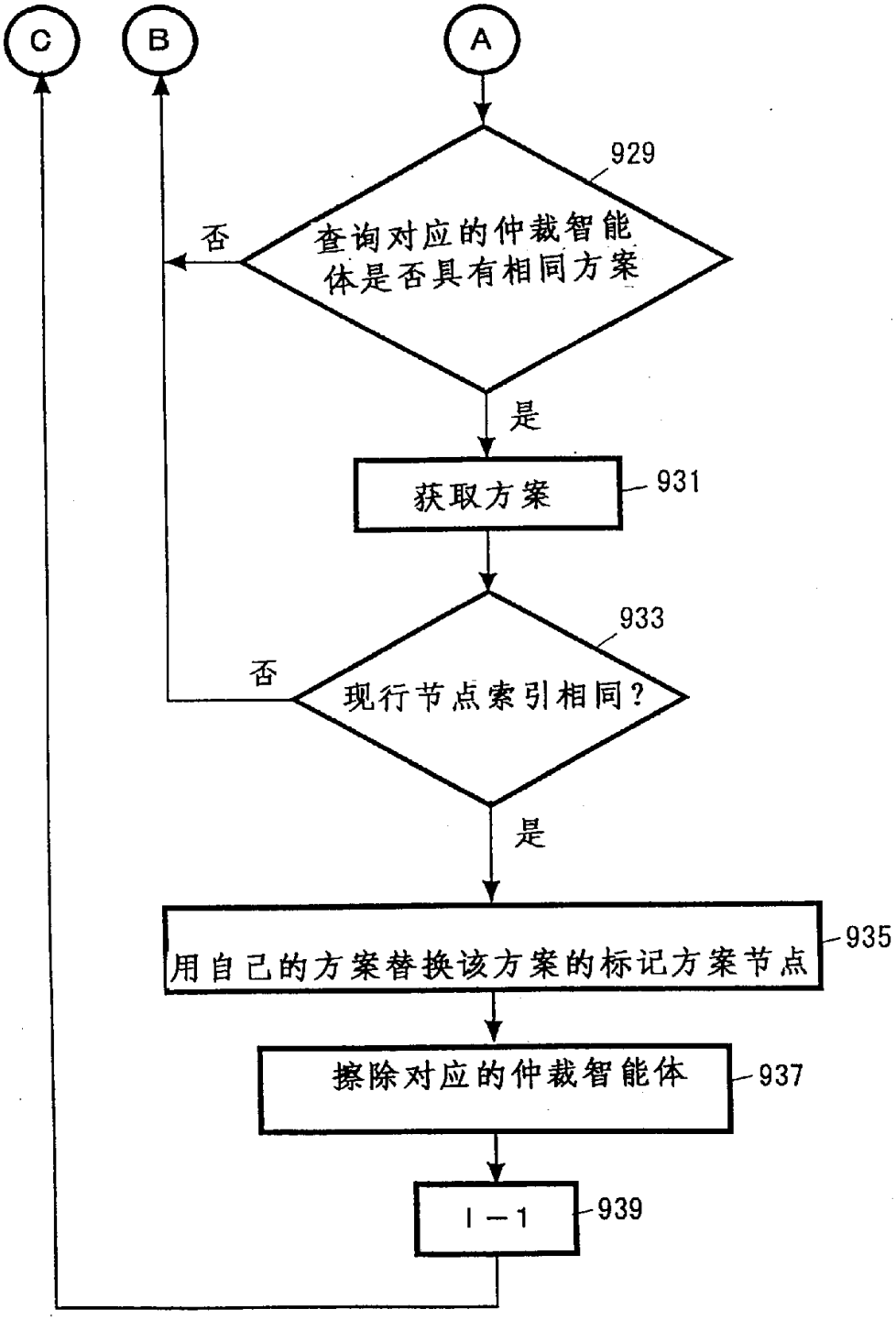

C
B
A
929
查询对应的仲裁智能体是否具有相同方案
否
是
获取方案
931
933
现行节点索引相同？
否
是
用自己的方案替换该方案的标记方案节点
935
擦除对应的仲裁智能体
937
I－1
939

图35

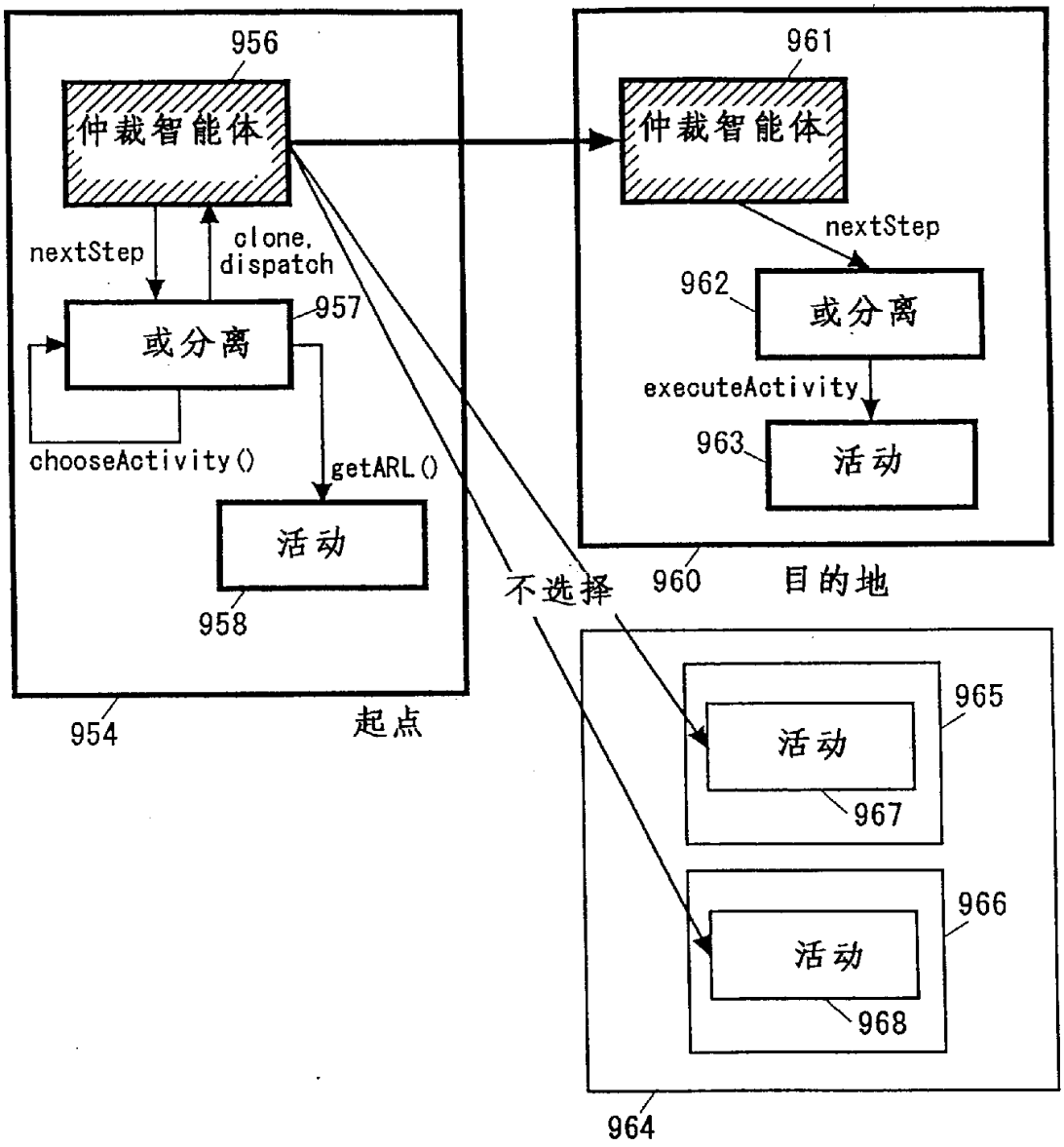

956
仲裁智能体
nextStep
clone,
dispatch
或分离
957
chooseActivity()
getARL()
活动
958
954
起点
961
仲裁智能体
nextStep
962
或分离
executeActivity
963
活动
960
目的地
不选择
965
活动
967
966
活动
968
964

图36

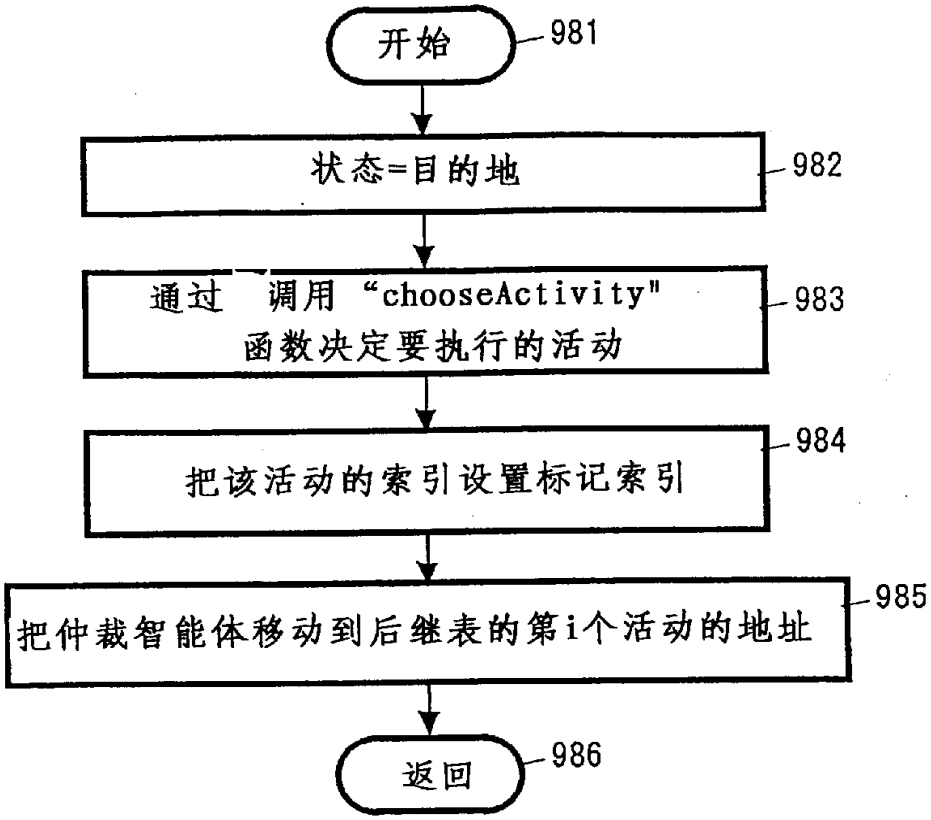

开始
981
状态=目的地
982
通过 调用 “chooseActivity" 函数决定要执行的活动
983
把该活动的索引设置标记索引
984
把仲裁智能体移动到后继表的第i个活动的地址
985
返回
986

图 37

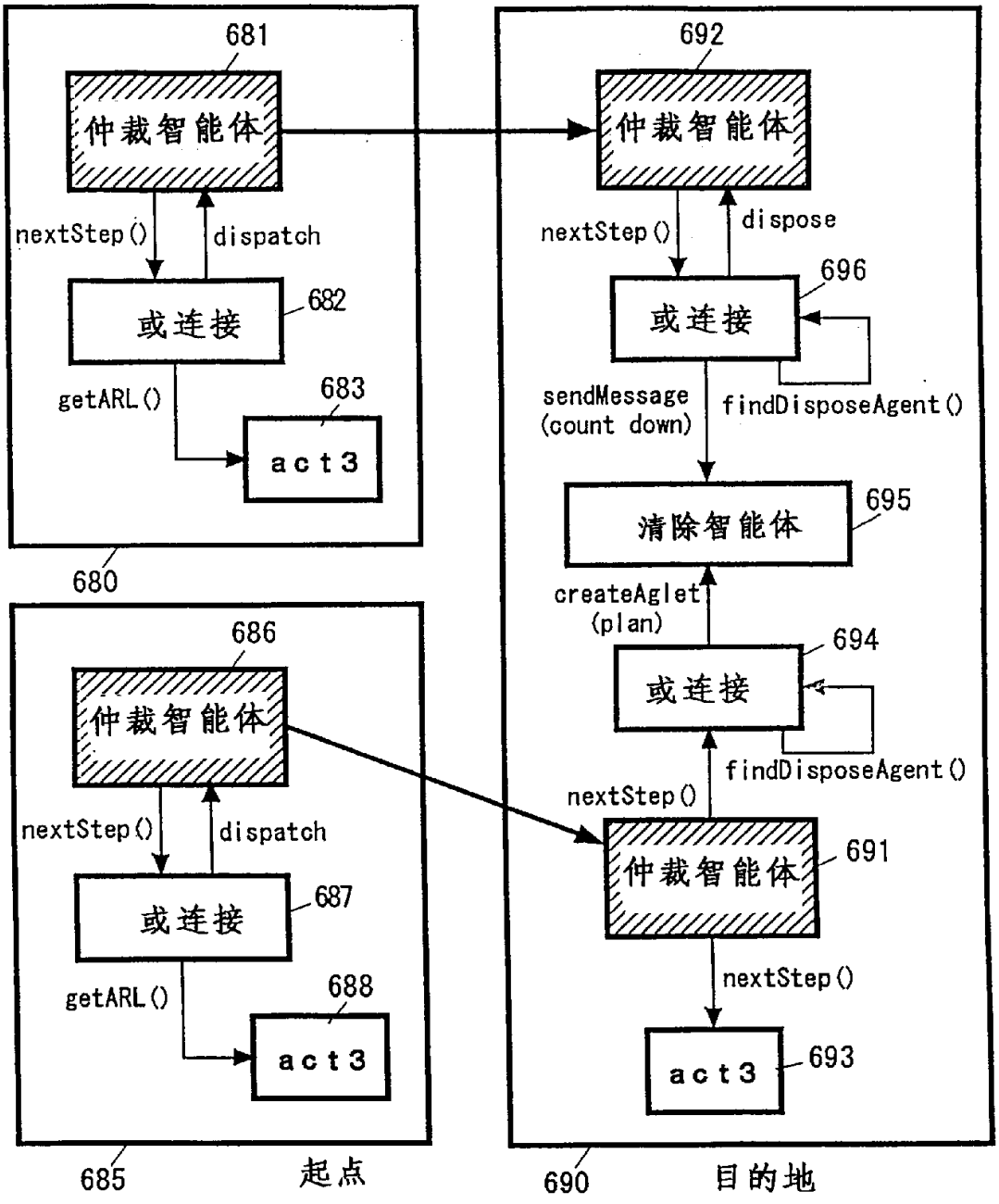
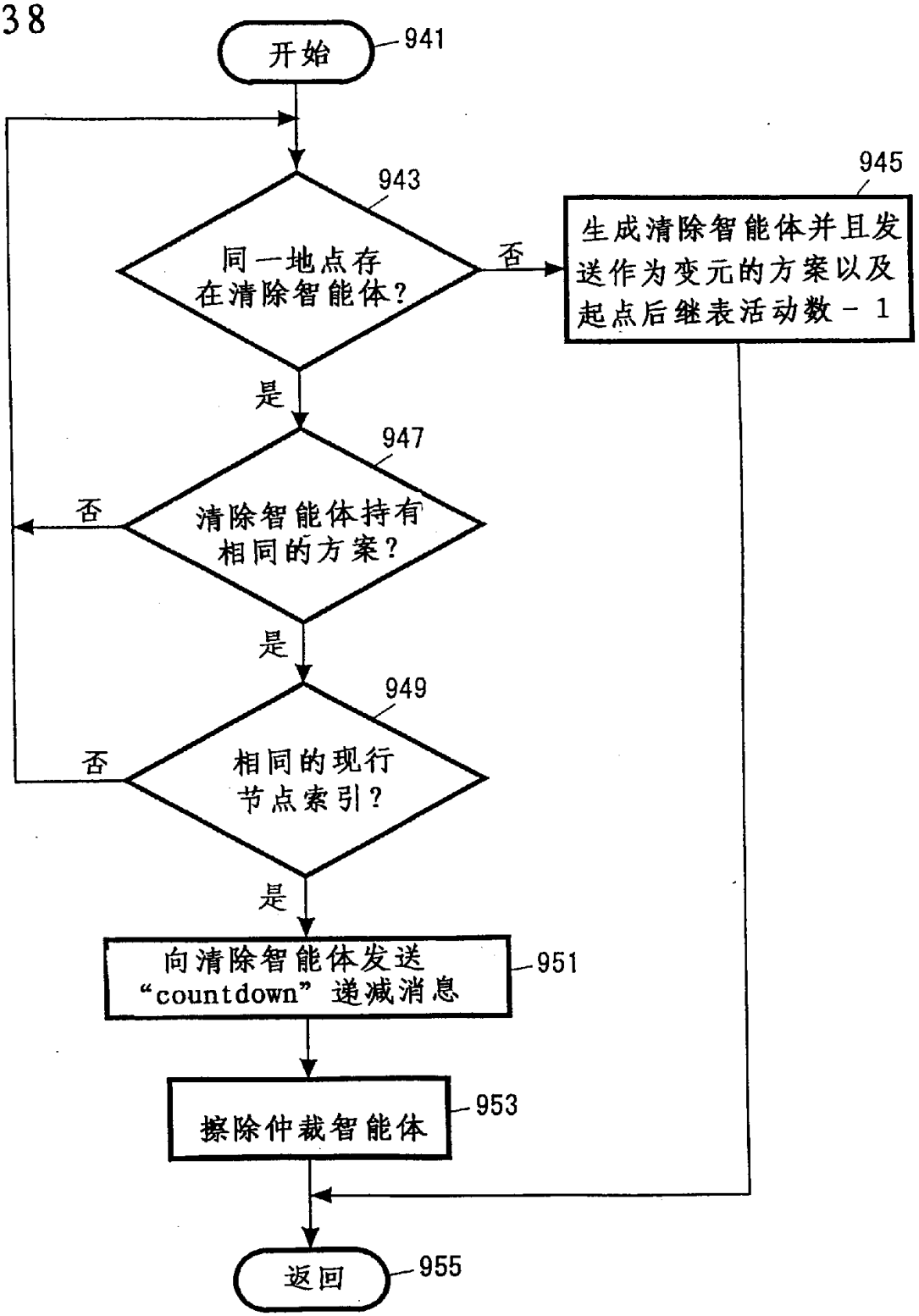

681
仲裁智能体
nextStep()
dispatch
或连接
682
getARL()
683
act3
680
686
仲裁智能体
nextStep()
dispatch
或连接
687
getARL()
688
act3
685
起点
692
仲裁智能体
nextStep()
dispose
696
或连接
sendMessage
(count down)
findDisposeAgent()
695
清除智能体
createAglet
(plan)
694
或连接
findDisposeAgent()
nextStep()
691
仲裁智能体
nextStep()
693
act3
690
目的地

图38


开始
941
943
同一地点存在清除智能体?
否
945
生成清除智能体并且发送作为变元的方案以及起点后继表活动数－1
是
947
清除智能体持有相同的方案?
否
是
949
相同的现行节点索引?
否
是
向清除智能体发送“countdown”递减消息
951
擦除仲裁智能体
953
返回
955