(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 December 2007 (13.12.2007)

PCT

(10) International Publication Number
WO 2007/143265 A2

(51) International Patent Classification:
G06F 17/10 (2006.01)

(21) International Application Number:
PCT/US2007/065574

(22) International Filing Date: 30 March 2007 (30.03.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/421,012 30 May 2006 (30.05.2006) US

(71) Applicant *(for all designated States except US)*: **MOTOROLA, INC.** [US/US]; 1303 East Algonquin Road, Schaumburg, Illinois 60196 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **THOMPSON, William, K.** [US/US]; 550 Sheridan Square Apartment 5A, Evanston, Illinois 60202 (US). **DAVIS, Paul, C.** [US/US]; 3922 N. Tripp Avenue, #2, Chicago, Illinois 60641 (US).

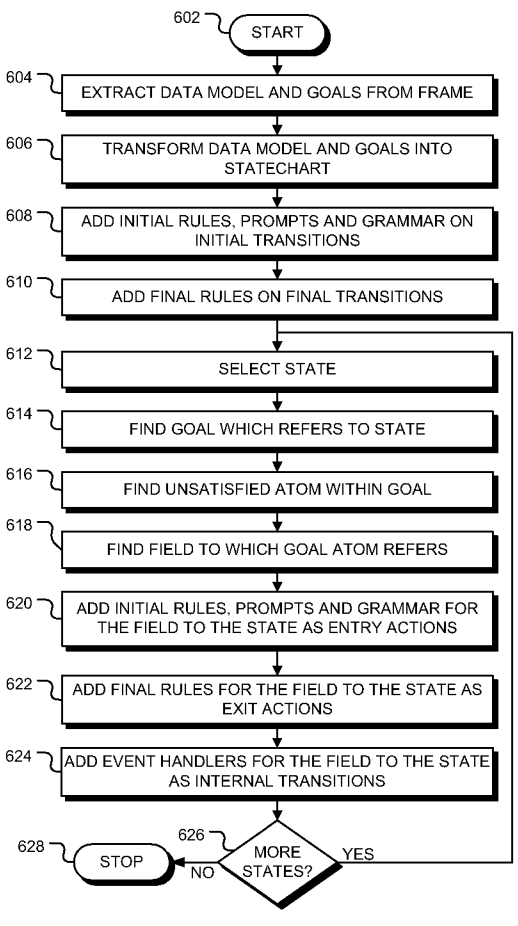
(74) Agents: **LAMB, James, A.** et al.; 1303 East Algonquin Road, Schaumburg, Illinois 60196 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: STATECHART GENERATION USING FRAMES



(57) Abstract: A Statechart (400) for management of an interaction is generated automatically from a frame (100, 200, 304) that describes the interaction by extracting a data model (300) and a set of goals (302) from the frame and then generating a Statechart (400) from the data model (300) and the set of goals (302). The data model (300) includes a set of data fields (306, 308) to be completed during the interaction. Procedural logic (206, 208) is extracted from the frame and used to annotate the Statechart with, for example, initial and final rules, prompts and grammar (502-512). The Statechart (400) may be stored for future use or editing, or used to manage a current interaction.

WO 2007/143265 A2

Published:

— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

STATECHART GENERATION USING FRAMES

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is related to co-pending U.S. patent applications docket number CML03559HI, titled "Hierarchical State Machine Generation from Interaction Management Using Goal Specification," and docket number CML03562HI, titled "Frame Goals for Dialog System," filed concurrently with the present application, both filed even date with this application.

BACKGROUND

[0002] Many techniques currently exist for specifying dialogue control logic in voice and multimodal dialogue systems. At the lowest level of abstraction are finite state machines (FSMs), which explicitly enumerate the various states and transitions in a dialog flow. FSMs have frequently been used in as a technique for specifying dialog flows. Recently, proposals have been made to use Harel Statecharts (Statecharts), also known as Hierarchical State Machines (HSMs), as a generic control language for specifying user interaction control logic. Statecharts are similar to FSMs, but they are augmented with a variety of additional constructs, including hierarchical states, guard conditions, and parallel states. These added constructs can make Statecharts simpler and more extensible than equivalent FSMs, because they factor out common behavior into common super-states, eliminating duplication of logic.

[0003] At a higher level of abstraction than FSMs and Statecharts are frame-based techniques for dialog management. In many task-based dialogs, a system requires certain pieces of information from the user in order to accomplish some domain specific task (such as booking a flight, finding a restaurant, or finding out who you want to call on the phone). A frame is a data structure that can hold the required information. A primary advantage of frame-based techniques over finite-state scripts is that they enable a dialog designer to create a relatively complex dialog in a more compact format. A frame succinctly represents a large number of states by eliminating much of the explicit process logic that is required in FSMs and Statecharts. Another advantage of frame-based techniques is that it is easier to model mixed-initiative dialog, because it is easy to specify grammars, prompts, and actions that have scope over multiple fields contained within a form. A primary reason for the current popularity of frames is the existence of standards, such as the World Wide Web Consortium Voice Extensible Markup Language version 2.0 (W3C VoiceXML 2.0) standard, which adopt the frame-based approach to dialog specification. In the VoiceXML 2.0 standard, frames come in two varieties, "forms", and "menus". An example of a VoiceXML frame is shown in **FIG 1**. An example of a frame written in the frame-specification language Motorola Portable Dialog Frame Language (MPD-FL) of the Motorola Corporation is shown in **FIG 2**.

[0004] While there are some advantages to frame-based techniques for dialog specification, there are some disadvantages as well. These stem from the fact that frame-based dialog managers require built-in algorithms for interpreting frames, since the frame itself is a primarily declarative structure that omits most of the process control logic required to use it in a dialog. In VoiceXML, this built-in algorithm is

called the "Form Interpretation Algorithm" (FIA). In this document, the term "FIA" is used as a generic term for any algorithm that reads in a frame and generates a corresponding dialog flow. This reliance on an FIA leads to two sorts of problems: Firstly, it can be hard to verify and debug a frame, since it isn't easy to visualize the current state and the current possibilities for transitioning to other states. Secondly, if the dialog designer wants to create a dialog that doesn't fit well with the built-in FIA, then he or she must struggle against the constraints of the framework in order to implement the desired logic.

[0005] While the first problem could be remedied to some degree with proper visualization tools, the second problem is intrinsic to the use of frames. It remains the case that frame-based techniques are suitable for some kinds of dialogs (those fitting well with the "form-filling" or "menu selection" metaphors) but not for many other types of dialog. This has prompted designers to look at the use of Statecharts as a control language for the future VoiceXML 3.0 standard. This new control language has been termed Statechart XML (SCXML) by the W3C Working Group, which plans on using it to augment the frame-language defined in VoiceXML 2.0.

[0006] The technical details of Statecharts are known in the art and use of Statecharts with dialog systems has already been proposed. However, prior publications do not describe how to generate the Statecharts from higher-level dialog abstractions. The generation of deterministic Statecharts from feature models has been disclosed, but features models are quite different from frames and frame constructs. Feature models are more like domain models than compact descriptions of possible dialog moves, such as forms and menus.

[0007] Statecharts have also be used as a starting point for generating software, but automatic generation of the Statecharts themselves from other constructs has not been proposed. In addition, the use of declarative constructs (other than frame constructs) to generate simple state machines has been disclosed, but these do not generate Statecharts.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as the preferred mode of use, and further objects and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawing(s), wherein:

[0009] **FIG 1** shows an example of a frame specified using the VoiceXML language.

[0010] **FIG. 2** shows an example of a frame specified using the Motorola Portable Dialog Frame Language (MPD-FL) of the Motorola Corporation.

[0011] **FIG. 3** is a diagram showing generation of a data model and goals from an exemplary frame consistent with certain embodiments of the invention.

[0012] **FIG. 4** is a diagram showing generation of a Statechart from a data model and goals consistent with certain embodiments of the invention.

[0013] **FIG. 5** is a diagram showing the addition of procedural logic to a Statechart consistent with certain embodiments of the invention.

[0014] **FIG. 6** is a flow chart of a method, consistent with certain embodiments of the invention, for generating a Statechart from a frame.

DETAILED DESCRIPTION

[0015] While this invention is susceptible of embodiment in many different forms, there is shown in the drawings and will herein be described in detail one or more specific embodiments, with the understanding that the present disclosure is to be considered as exemplary of the principles of the invention and not intended to limit the invention to the specific embodiments shown and described. In the description below, like reference numerals are used to describe the same, similar or corresponding parts in the several views of the drawings.

[0016] One embodiment of the present invention relates to a combination of Harel Statecharts (Statecharts) (also known as Hierarchical State Machines (HSMs)) and Frames in a single interaction management framework (IMF). This combination aids interaction designers, especially for dialog-centric parts of the design, by automatically generating a Statechart (or portions of a Statechart) from higher-level dialog constructs, such as forms, menus, and the like. The automatically generated Statechart can subsequently be iteratively fleshed out and modified by the designer, which is an easier task than starting a new Statechart from scratch.

[0017] In accordance with one embodiment, frame constructs, including MPD-FL and VoiceXML forms and menus, are mapped into equivalent Statecharts. This mapping from frames to Statecharts essentially creates semantic representation of frames in terms of Statecharts. Given this semantic representation, it becomes possible to seamlessly combine dialogs specified as frames with dialogs specified directly as Statecharts.

[0018] Interaction designers can use dialog-specific constructs (also referred to as abstractions), such as forms and menus, to generate Statecharts components (sets of states and transitions). Such built-in constructs, at both the tool-level and markup-level, simplify the design process, since notions such as "select one of the options" and "fill in all the slots" need not be manually entered but can be automatically generated from a simpler specification.

[0019] This interaction/dialog abstraction can be also be extended to any number of interaction constructs (beyond forms and menus), to other common constructs (e.g., give feedback, get help, restart, select two, enter user info, etc), and more importantly, to designer-defined abstractions which can be stored in a library and reused.

[0020] It is noted that this approach is distinct from the concept of storing a portion or all of a Statechart in a file and reusing it. The distinction is that the abstraction (the concept) is being stored for reuse, perhaps to be selected from a pull-down menu or button-click at the tool-level. While this will lead to Statecharts being stored and reloaded, it makes the design process easier, since again the designer can focus on a concept that is much simpler than the states and transitions generated from the concept.

[0021] The exemplary embodiment, disclosed below with reference to **FIG. 3**, is described using MPD-FL terminology, but it will be apparent to those of ordinary skill in the art that other terminology could be used, such as using VoiceXML constructs,

[0022] **FIG. 3** is a diagram showing generation of a data model and goals from an exemplary frame consistent with certain embodiments of the invention. Referring to **FIG. 3**, data model 300 and goals 302 are extracted from a given frame 304. In this

example, the data model 300 includes a phonebook 306 having fields labeled 'cmd' and 'entry'. The 'cmd' field is atomic, and takes string values corresponding to "add", "delete", "display", and the like. The 'entry' field is non-atomic, in that its value is an embedded data structure that includes its own set of fields, consisting of 'fname', 'lname' and 'number' shown in box 308, These correspond to the first name, last name and telephone number of a phonebook entry. The data model 300 and goals 302 are derived from the declarative portion of the frame 304. The first goal (G1) is that the command field (cmd) and the entry field (entry) in the data model have both been filled. The second goal (G2) is that both the first and last name fields have been filled, or that the telephone number field has been filled Goal G1 is satisfied only if goal G2 is satisfied as well, since the data structure which G2 refers to is embedded inside of the data structure that G1 refers to. In general, the data model is structured as a tree, where each node, consisting of a set of fields, has its own goal. There will therefore be exactly one goal specification per node of the data model tree. In general, we represent goals as Boolean expressions over fields, and we express these goals in Disjunctive Normal Form (disjunctions of conjunctions). In what follows, the field expressions in goals are referred to as "atoms". These are predicates over fields which evaluate to true if and only if the field referred to is set to a single legitimate value.

[0023] **FIG. 4** is a diagram showing generation of a Statechart from a data model 300 and set of goals 302 consistent with certain embodiments of the invention. The resulting Statechart 400 contains a set of states, each representing a state of completion of the associated data model. The term 'completion' is taken to include full completion (all fields filled) and partial completion (some fields filled) as well as

no completion (no field filled). The phonebook Statechart 402 includes an initial pseudo-state 404, a state 406 corresponding to the empty set, states 408 and 410 for which a single field of the data model is filled, a state 412 for which both fields of the data model are filled and a final state 414. Each transition in the Statechart, denoted by an arrow connecting two states, represents a change in the state of the associated data model. The entry Statechart 416 includes an initial pseudo-state 418, a state 420 corresponding to the empty set, states 422, 424 and 426 for which a single field of the data model is filled, states 428, 430 and 432 for which two fields of the data model are filled, state 434 for which all fields are filled, and a final state 436. In Statechart 416, the letter 'l' is used to denote the last name field (lname), the letter 'f' is used to denote the first name field (fname), the letter 'n' is used to denote the number field (number). The Statechart is generated by first generating the set of states corresponding to completion states of the data model may be generated, together with an initial pseudo-state and a final state. A transition is generated from the initial pseudo-state to an empty state of the set of states and further transitions are generated between each state of the set states and its subset states. Outgoing transitions from any state of the set of states that satisfies at least one goal of the set of goals are removed (shown by broken lines in Statechart 416) and transitions to the final state are generated from these states.

[0024] **FIG. 5** is a diagram showing the addition of procedural logic to an exemplary Statechart consistent with certain embodiments of the invention. The procedural logic is added to the Statechart as guards and actions on transitions, and as entry and exit actions for states. Initial rules, prompts and grammar 502 as specified in the frame, are added to the Statechart on the initial transition, as actions. Rule pre-

conditions are added as guards. Final rules 504 for the frame are added to the Statechart on every final transition, as actions. Again, rule pre-conditions are added as guards. Fields are handled as follows: For each state **s** of a Statechart **H**, the first goal **g** which refers to **s** is found. Within the goal **g**, the first unsatisfied atom **a** is found. The initial rules, prompts and grammar of the field **f** (to which the atom **a** refers) are added to the state **s** as entry actions. For example, when control of the interaction transitions to the {cmd} state 410 in **FIG. 5**, the goal G1 in **FIG. 3** indicates that the 'entry' field must be filled to satisfy the goal. The initial rules, prompts and grammar 506 for the 'entry' field are thus added to the {cmd} state as initial actions. When {cmd} state is first entered, these initial rules will be used to prompt the user for the information required to reach the goal G1. Corresponding rule pre-conditions are added as guards. Similarly, when control transitions to the {entry} state 408 in **FIG. 5**, the goal G1 in **FIG. 3** indicates that the 'cmd' field must be filled to satisfy the goal. The initial rules, prompts and grammar 508 for the 'cmd' field are thus added to the {entry} state 408. When the {entry} state is first entered, these initial rules will be used to prompt the user for the information required to reach the goal G1. This process is repeated with the final rules of **f**, except these are added to **s** as exit actions (510 and 512 for example) and corresponding rule pre-conditions are added as guards. The final rules indicate actions to be taken following user input. This process is repeated again with the event handlers of **f**, except these are added to the state **s** as internal transitions, such as 512, labeled by the event handled. The rules of the event handlers, as specified in the frame, are also added to the internal transitions in the Statechart, together with corresponding rule pre-conditions (added as guards). For example, if a user input does not satisfy the final rules, control may

transition back the current state, as indicated by transition 514.

[0025] **FIG. 6** is a flow chart of a method, consistent with certain embodiments of the invention, for generating a Statechart from a frame. Following start block 602 in **FIG. 3**, a data model *D* and goals *G* are extracted from a given frame *F* at block 604. The data model *D* and goals *G* are derived from and consist of the declarative portion of the frame *F*. At block 606, data model *D* and goals *G* are transformed into a Statechart (HSM) *H*, as described in the co-pending application docket number CML03559HI. The remaining blocks of **FIG. 6** describe the annotation of the Statechart *H* with procedural logic from the frame *F*. At block 608, initial rules, prompts and grammar of *F* are added to *H* on the initial transition, as actions. Corresponding rule pre-conditions may also be added as guards. At block 610, final rules of *F* are added to *H* on every final transition, as actions, and any rule pre-conditions are added as guards. At block 612 a state **s** of the Statechart is selected. The first goal **g** which refers to **s** is found at block 614 and, at block 616, the first unsatisfied atom **a** is found within the goal **g**. At block 620, the initial rules, prompts and grammar of field **f** (to which **a** refers) are added to **s** as entry actions and any corresponding rule pre-conditions are added as guards. This process is repeated at block 622 with the final rules of **f**, except these are added to **s** as exit actions. Again, any rule pre-conditions are added as guards. At block 624, the process is repeated with the event handlers of the frame **f**, except these are added to the state **s** as internal transitions, labeled by the event handled. Finally, the rules of the event handlers are added to the internal transitions and any rule pre-conditions are added as guards. At decision block 626, a check is made to determine if there are any more states in the Statechart. If there are more states, as depicted by the positive branch from decision

block 626, flow returns to block 612, where a next state is selected. If there are no more states, as depicted by the negative branch from decision block 626, the process terminates at block 628.

[0026] The Statechart generated by this process may be stored in a suitable computer readable medium for current or future use, or for editing by an interaction designer. The Statechart may be combined with Statecharts generated by other means (either manual or automated).

[0027] The present invention, as described in embodiments herein, may be implemented by system having a programmed processor for executing programming instructions that are broadly described above in flow chart form that can be stored on any suitable electronic storage medium. The system may also include an input for receiving the data model and associated goals and an output for transmitting the generated Statechart to a remote location or a memory. However, those skilled in the art will appreciate that the processes described above can be implemented in any number of variations and in many suitable programming languages without departing from the present invention. For example, the order of certain operations carried out can often be varied, additional operations can be added or operations can be deleted without departing from the invention. Error trapping can be added and/or enhanced and variations can be made in user interface and information presentation without departing from the present invention. Such variations are contemplated and considered equivalent.

[0028] While the invention has been described in conjunction with specific embodiments, it is evident that many alternatives, modifications, permutations and variations will become apparent to those of ordinary skill in the art in light of the

foregoing description. Accordingly, it is intended that the present invention embrace all such alternatives, modifications and variations as fall within the scope of the appended claims.

What is claimed is:

What is claimed is:

1. A method for automatic generation of a Statechart for management of an interaction, the method comprising:

extracting a data model and a set of goals from a frame that describes the interaction, the data model comprising a set of data fields to be completed during the interaction;

generating a Statechart from the data model and the set of goals;

extracting procedural logic from the frame;

adding the procedural logic to the Statechart; and

outputting the Statechart.

2. A method in accordance with claim 1, wherein generating a Statechart from the data model and the set of goals comprises:

generating a set of states corresponding to completion states of the data model;

generating an initial pseudo-state and a final state;

generating a transition from the initial pseudo-state to an empty state of the set of states;

generating transitions between each state of the set states and its subset states;

removing outgoing transitions from any state of the set of states that satisfies at least one goal of the set of goals; and

generating transitions to the final state from any state of the set of states that satisfies at least one goal of the set of goals.

3. A method in accordance with claim 1, wherein adding the procedural logic to the Statechart comprises, for each state of the Statechart:

finding a goal of the set of goals that refers to the state;

finding an unsatisfied atom within the goal;

finding a field of the data model to which the goal atom refers;

adding initial rules, prompts and grammar for the field to the state as entry actions; and

adding final rules for the field to the state as exit actions.

4. A method in accordance with claim 3, wherein adding the procedural logic to the Statechart further comprises adding initial rule pre-conditions to the state as guards on the entry actions and adding final rule pre-conditions to the state as guards on the final actions.

5. A method in accordance with claim 3, wherein adding the procedural logic to the Statechart further comprises:

adding initial rules, prompts and grammar on initial transitions of the Statechart; and

adding final rules on final transitions of the Statechart.

6. A method in accordance with claim 3, wherein adding the procedural logic to the Statechart further comprises adding event handlers for the field to the state as internal transitions.

7. A method in accordance with claim 1, wherein the Statechart comprises:

a set of states corresponding to completion states of the data model;

an initial pseudo-state;

a final state;

a transition from the initial pseudo-state to an empty state of the set of states;

incoming transitions to each state of the set states from any of its subset states that do not satisfy any goal of a set of goals; and

outgoing transitions to the final state from any state of the set of states that satisfies at least one goal of the set of goals.

8. A computer readable medium storing programming instructions that, when executed on a processor, perform the method of claim 1.

9. A method in accordance with claim 1, wherein the frame comprises a representation stored on a computer readable representation.

10. A method in accordance with claim 1, further comprising presenting the Statechart to an interaction designer for modification.

11. A method in accordance with claim 1, further comprising combining the Statechart with a Statechart generated by a different method.

12. A system for automatic generation of a Statechart for management of an interaction, the system comprising:

an input for receiving a frame that describes the interaction;

a processor operable to:

extract a data model and a set of goals from a frame that, the data model comprising a set of data fields to be completed during the interaction;

generate a Statechart from the data model and the set of goals;

extract procedural logic from the frame; and

add the procedural logic to the Statechart; and

an output for outputting the Statechart.

13. A system in accordance with claim 12, further comprising a memory for storing the Statechart.

14. A computer readable medium storing a program of instructions that, when executed on a processor, generate a Statechart for management of an interaction, the program of instructions comprising:

instructions to extract a data model and a set of goals from a frame that, the data model comprising a set of data fields to be completed during the interaction;

instructions to generate a Statechart from the data model and the set of goals;

instructions to extract procedural logic from the frame;

instructions to add the procedural logic to the Statechart; and

instructions to output the Statechart.

100

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.w3.org/2001/vxml
   http://www.w3.org/TR/voicexml20/vxml.xsd">
<form>
  <block>
    We need a few more details to complete your order.
  </block>
  <field name="color">
    <prompt>Which color?</prompt>
    <option>red</option>
    <option>blue</option>
    <option>green</option>
  </field>
  <field name="size">
    <prompt>Which size?</prompt>
    <option>small</option>
    <option>medium</option>
    <option>large</option>
  </field>
  <field name="quantity">
    <grammar type="application/srgs+xml" src="/grammars/number.grxml"/>
    <prompt>How many?</prompt>
  </field>
  <block>
    Thank you.  Your order is being processed.
    <submit next="details.cgi" namelist="color size quantity"/>
  </block>
  <catch event="help nomatch">
    Your options are <enumerate/>.
  </catch>
</form>
</vxml>
```

FIG. 1

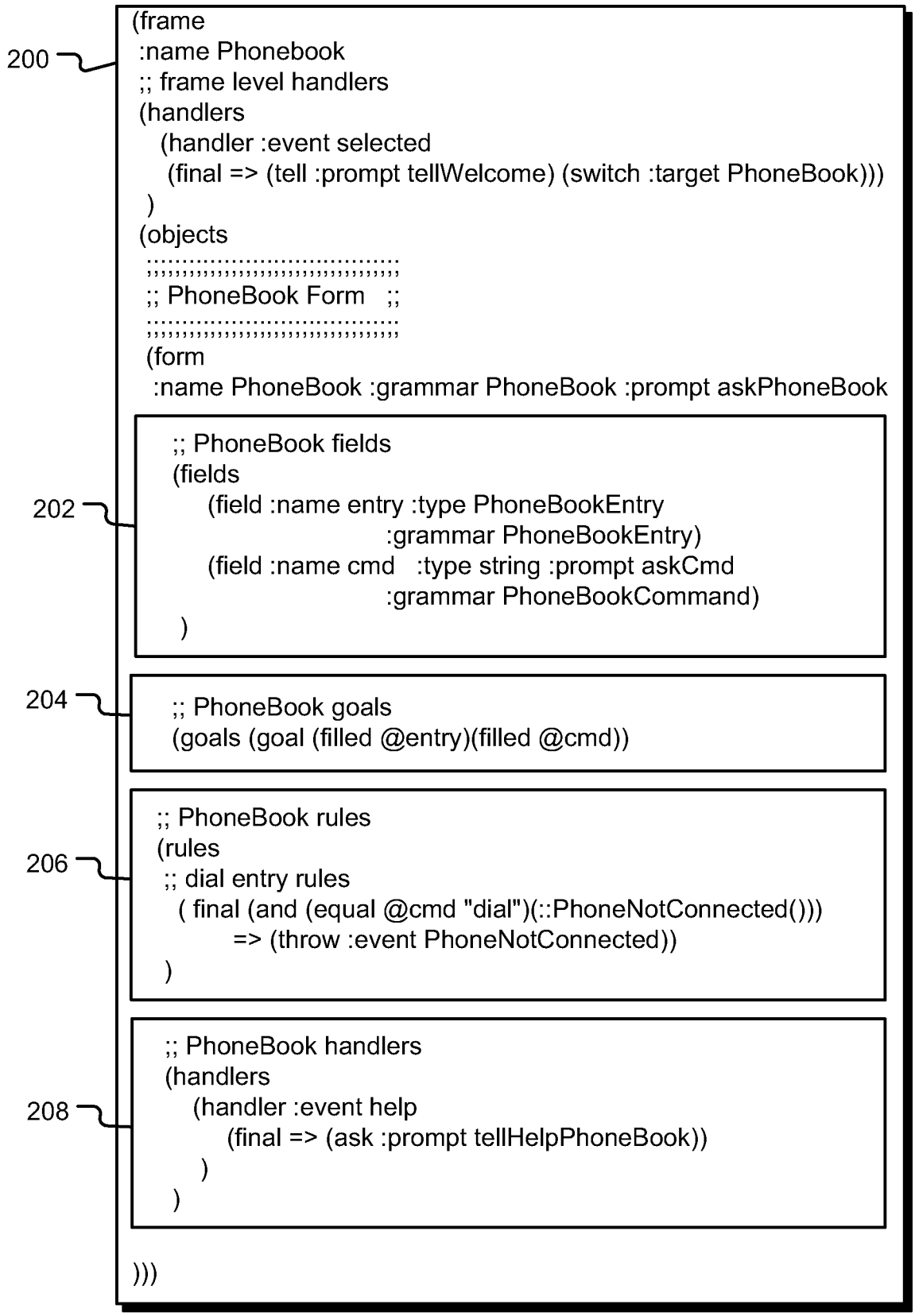

200
202
204
206
208
(frame
:name Phonebook
;; frame level handlers
(handlers
(handler :event selected
(final => (tell :prompt tellWelcome) (switch :target PhoneBook)))
)
(objects
;; PhoneBook Form ;;
(form
:name PhoneBook :grammar PhoneBook :prompt askPhoneBook
;; PhoneBook fields
(fields
(field :name entry :type PhoneBookEntry
:grammar PhoneBookEntry)
(field :name cmd :type string :prompt askCmd
:grammar PhoneBookCommand)
)
;; PhoneBook goals
(goals (goal (filled @entry)(filled @cmd))
;; PhoneBook rules
(rules
;; dial entry rules
(final (and (equal @cmd "dial")(::PhoneNotConnected()))
=> (throw :event PhoneNotConnected))
)
;; PhoneBook handlers
(handlers
(handler :event help
(final => (ask :prompt tellHelpPhoneBook))
)
)
)))

FIG. 2

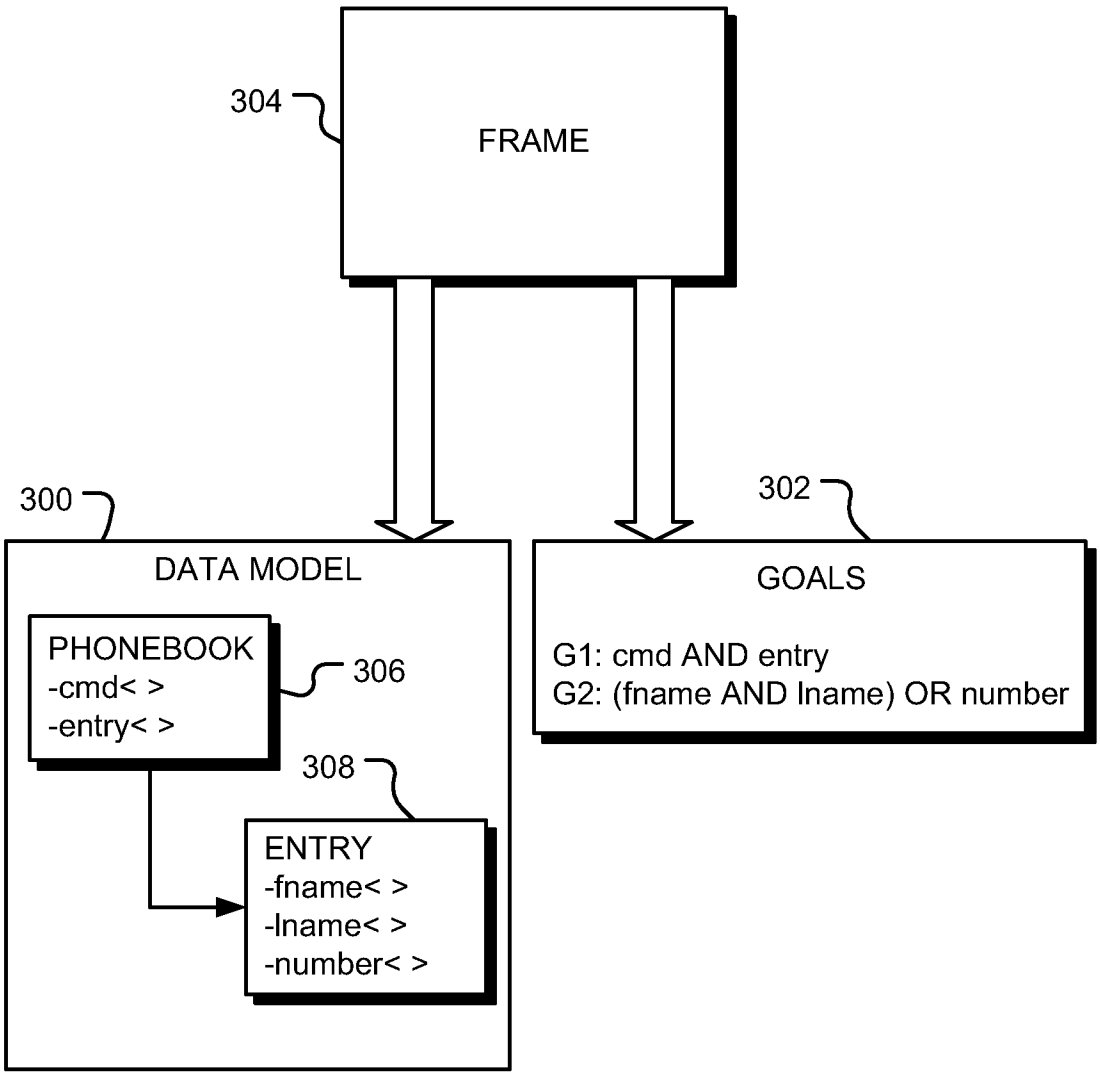

304
FRAME
300
DATA MODEL
PHONEBOOK
-cmd< >
-entry< >
306
308
ENTRY
-fname< >
-lname< >
-number< >
302
GOALS
G1: cmd AND entry
G2: (fname AND lname) OR number

FIG. 3

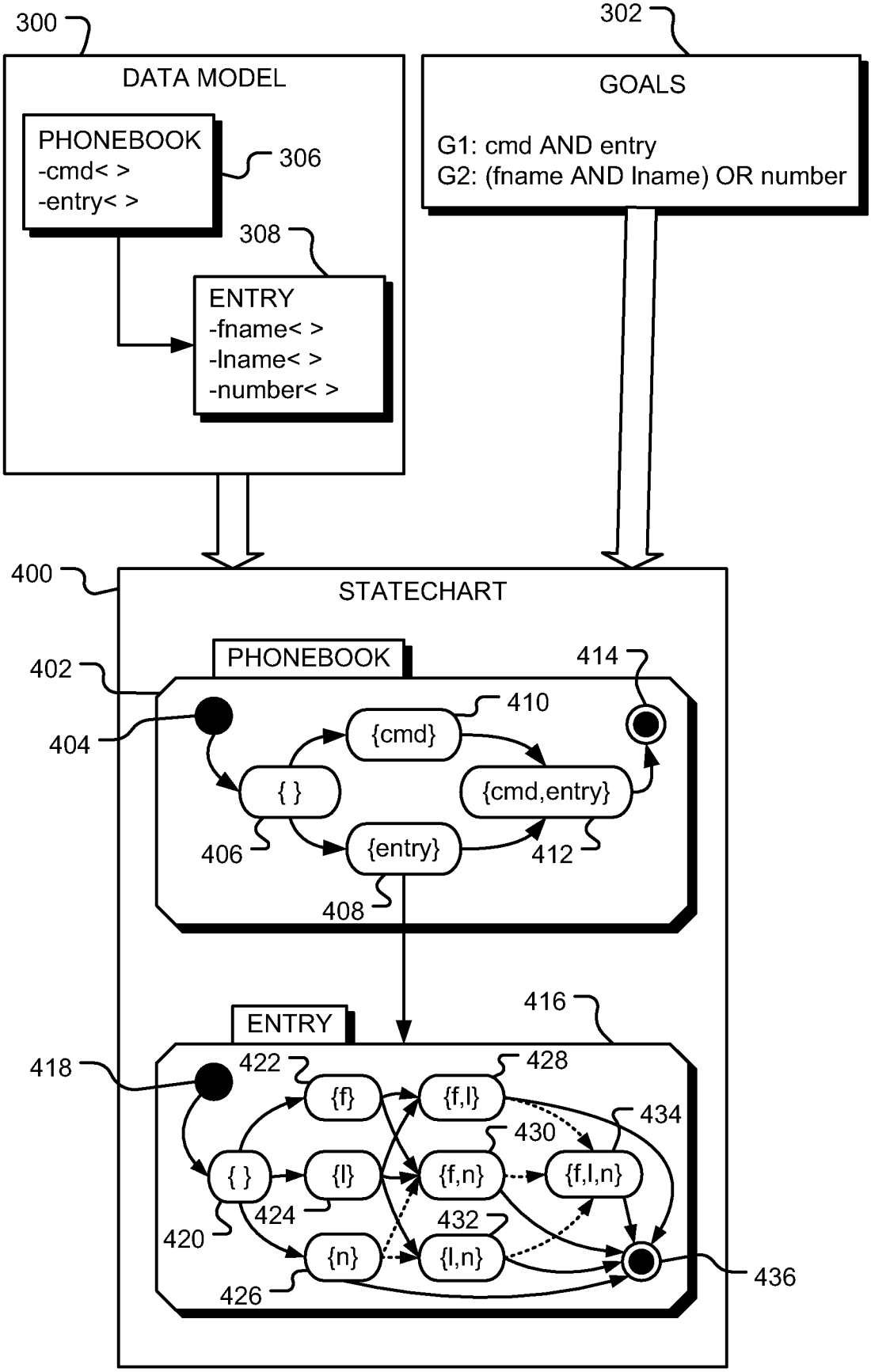

300
DATA MODEL
PHONEBOOK
-cmd< >
-entry< >
306
308
ENTRY
-fname< >
-lname< >
-number< >
302
GOALS
G1: cmd AND entry
G2: (fname AND lname) OR number
400
STATECHART
PHONEBOOK
402
404
414
410
{cmd}
{ }
{cmd,entry}
406
{entry}
412
408
ENTRY
416
418
422
428
{f}
{f,l}
434
430
{ }
{l}
{f,n}
{f,l,n}
424
420
432
{n}
{l,n}
436
426

FIG. 4

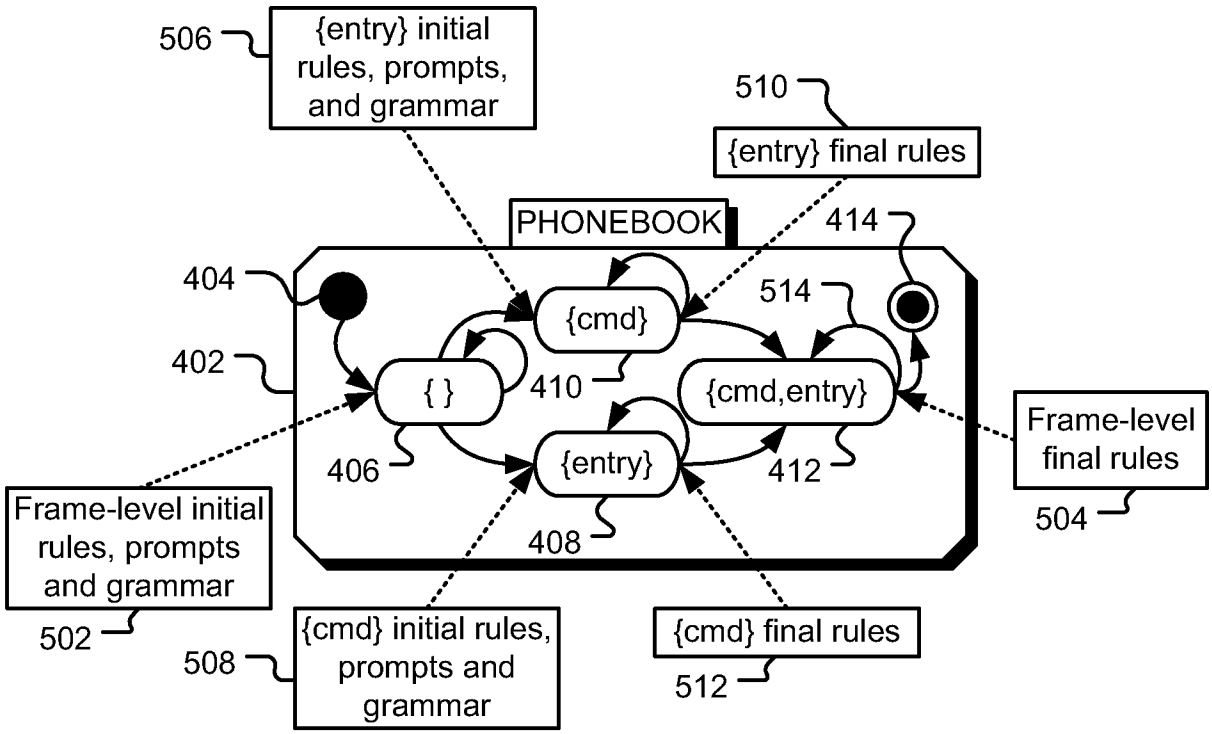

506
{entry} initial rules, prompts, and grammar
510
{entry} final rules
PHONEBOOK
414
404
514
{cmd}
402
{ }
410
{cmd,entry}
Frame-level final rules
406
{entry}
412
504
Frame-level initial rules, prompts and grammar
408
502
{cmd} initial rules, prompts and grammar
{cmd} final rules
508
512

FIG. 5

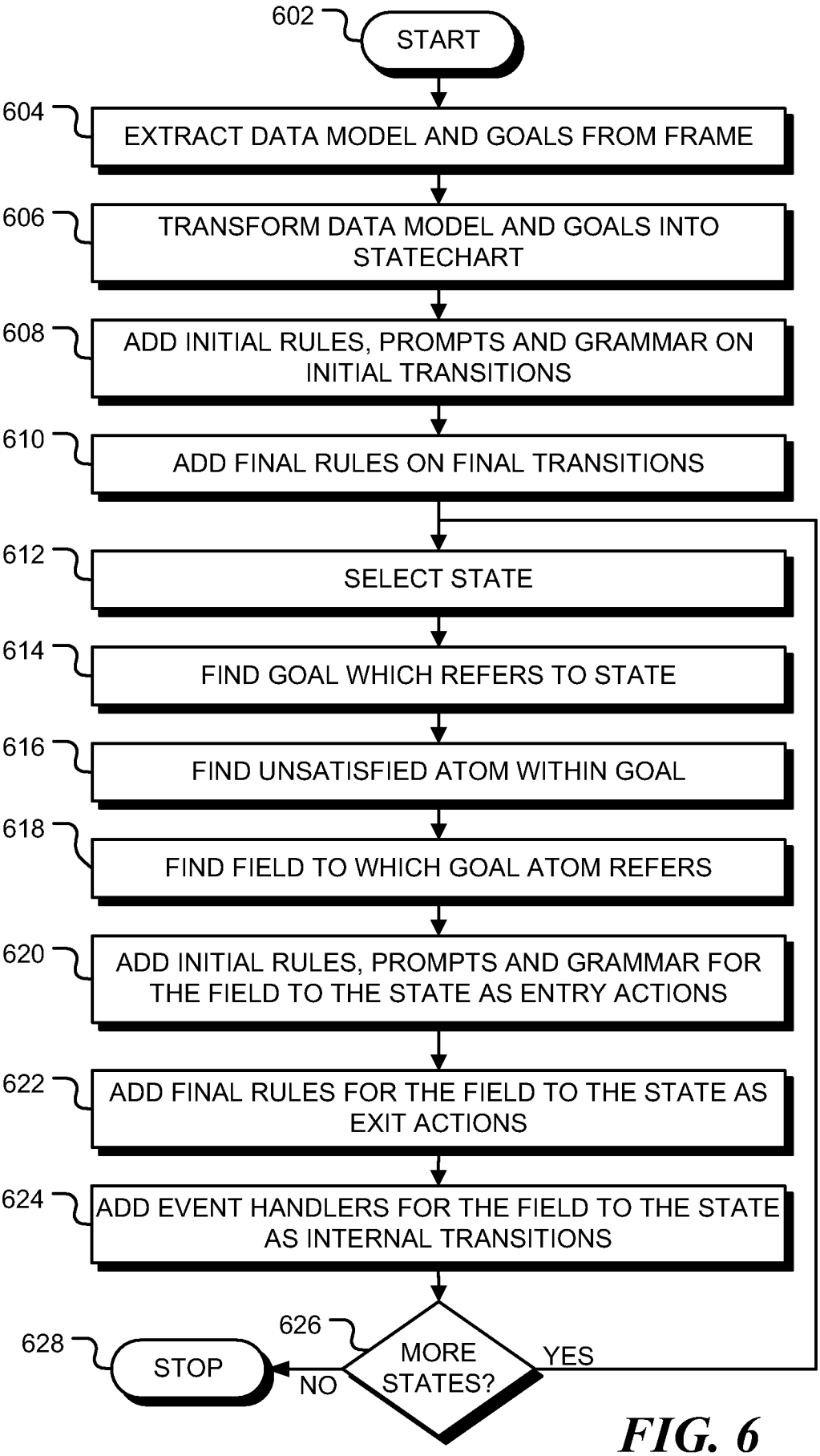

602
START
604
EXTRACT DATA MODEL AND GOALS FROM FRAME
606
TRANSFORM DATA MODEL AND GOALS INTO STATECHART
608
ADD INITIAL RULES, PROMPTS AND GRAMMAR ON INITIAL TRANSITIONS
610
ADD FINAL RULES ON FINAL TRANSITIONS
612
SELECT STATE
614
FIND GOAL WHICH REFERS TO STATE
616
FIND UNSATISFIED ATOM WITHIN GOAL
618
FIND FIELD TO WHICH GOAL ATOM REFERS
620
ADD INITIAL RULES, PROMPTS AND GRAMMAR FOR THE FIELD TO THE STATE AS ENTRY ACTIONS
622
ADD FINAL RULES FOR THE FIELD TO THE STATE AS EXIT ACTIONS
624
ADD EVENT HANDLERS FOR THE FIELD TO THE STATE AS INTERNAL TRANSITIONS
626
MORE STATES?
YES
NO
628
STOP

FIG. 6