

A Multiagent Planning Approach to Model a Tele-Surgery Domain

Amod Kumar Lal

Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, 247667 India
E-mail: akalpec@iitr.ernet.in

Rajdeep Niyogi

Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, 247667 India
E-mail: rajdpfec@iitr.ernet.in

Abstract— Technological advancements have led to the development of few commercially available tele-surgery systems till date. However such systems are very expensive. In tele-surgery, the task of a surgeon (the activities related to a surgery) is partially executed by a robot. Typically, the robot is under the control of a surgeon; it executes the instructions of the controlling surgeon. In this paper we give formal model of a tele-surgery domain (heart surgery) as a multiagent planning problem. The actions related to the surgery are represented as planning operators. The model consists of two interactive agents, referred to as EXPERT and INTERN. The EXPERT controls the activities of the INTERN. The INTERN executes the actions suggested by the EXPERT. The state space of each agent is modeled as a transition system. The communication of the agents is modeled using CCS. We have defined a condition to establish the success of the surgery using notions of finite games. We have also developed a prototype implementation incorporating the above features.

Index Terms — Tele Surgery, Agent Communication, CCS, Multiagent Planning, Bisimulation

I. Introduction

A simple example of a multiagent system given in [1] would be a doubles tennis match. In this game each team has two players (agents). The two players (in a team) are necessary for the game to be played. Some important features here include: joint action, coordination, and communication among the agents. A joint action in a tennis match would be ‘waiting near the net’ for one agent while the other agent ‘returns the ball’. Here the two individual actions of the agents occurring concurrently would constitute one primitive action for the team, which is often referred to as a joint action. In order that the joint action is successful the agents need to coordinate and communicate, which results in synchronization of the agents.

In tele-surgery [2], the objective is to perform a surgery remotely. That is, the surgeon and the patient are geographically apart. At the patient’s end a robot (instead of a human) performs the surgery. The robot is controlled by the surgeon, who is remotely located. The robot is expected to [faithfully] execute the instructions of the controlling surgeon. We may abstract this situation as a two-agent system. An important similarity of this system and the tennis domain is the feature of joint action. The robot waits till it has received an instruction from the surgeon. Here a joint action would include sending an instruction by the surgeon, performing the instruction by the robot upon receipt, and sending the resulting state back to the surgeon. (Modern systems may even avoid sending the resulting state, as the surgeon may observe the state remotely.) Thus ‘sending an instruction’ and ‘performing the instruction’ would constitute a joint action. A close look at this joint action and the one for the tennis match would reveal an important aspect on the roles of the agents. In the tennis domain the roles of the agents are interchangeable. This is not so in the tele-surgery domain; instruction would always be sent by the surgeon and the robot would always perform an instruction. Another crucial difference in these two domains is with respect to communication among agents. It is implicit in the tennis domain. It is explicit in the tele-surgery domain and it would be appropriate to reflect the communication in the joint action. Given this informal notion of a joint action, a description of the initial state, a description of the actions, and a goal state, a multiagent planning problem then would be to find a sequence of such joint actions that transform the initial state to the goal state. We model this idea on the heart surgery domain that we have chosen for this study.

Tele-surgery domains are quite complex. The complexity arises from several angles. Some among these include communication among the agents, controlling a robot, and medical imagery and transmission. Thus understanding such domains would call for expertise from several research areas like computer science, artificial intelligence, robotics, computer vision, and electronic communication

technologies. Most works in tele-surgery domains are primarily concerned on issues involving robotics [5,10] and on system development [3,4,7,9]. This work is an extended version of our earlier work [12]. We have included the following: (i) detailed description of the heart surgery domain—section II, (ii) modeling agent communication using CCS—section III C, (iii) defining the condition to express the success of the surgery—section III D, (iv) experimental results for handling recovery cases—section IV.

The rest of the paper is organized as follows. In section II we define the planning operators corresponding to the steps of a heart surgery. In section III we suggest a formal model of the heart surgery domain as a two agent system. In section IV we discuss the implementation details. The conclusions are given in section V.

II. Heart Surgery Domain

The human circulatory system [Fig. 1] is really a two-part system whose purpose is to bring oxygen-bearing blood to all the tissues of the body. When the heart contracts it pushes the blood out into two major loops or cycles. In the systemic loop, the blood circulates into the body's systems, bringing oxygen to all its organs, structures and tissues and collecting carbon dioxide waste. In the pulmonary loop, the blood circulates to and from the lungs, to release the carbon dioxide and pick up new oxygen. The systemic cycle is controlled by the left side of the heart, the pulmonary cycle by the right side of the heart.[6]

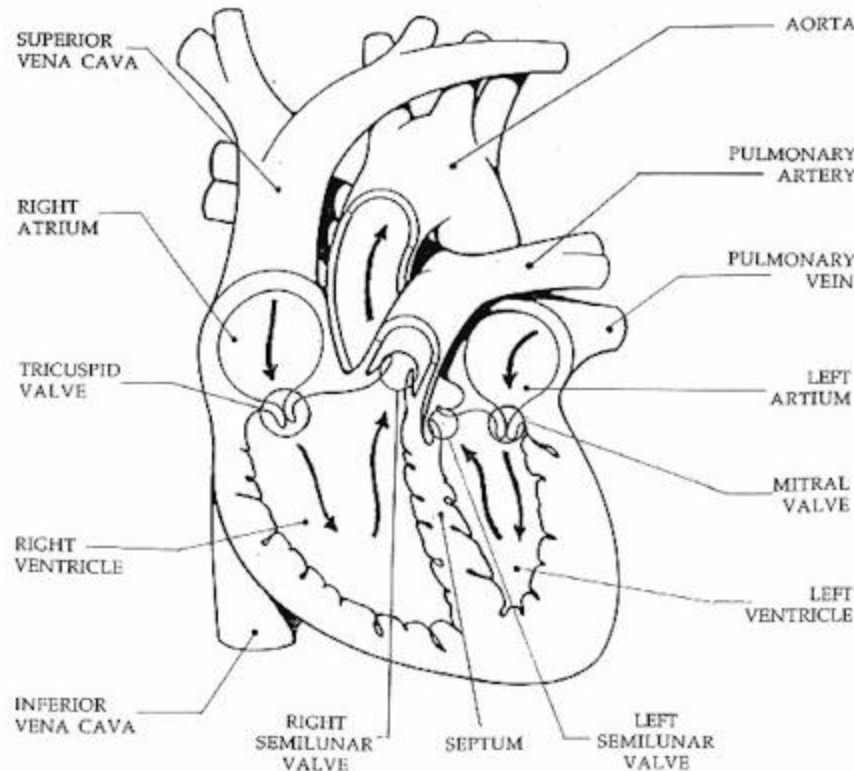


Fig. 1: The human circulatory system [6]

Superior vena cava is the big reason that causes Heart Bypass surgery. Superior vena cava (SVC) syndrome can present in a variety of ways, depending on the severity of the obstruction as well as the cause. Complete obstruction can cause a patient's condition to deteriorate rapidly. As a result, surgical bypass comes out as the quick solution for this.

The following steps are performed in surgery (Fig. 2):

Step01. Harvesting the saphenous vein: Choose and harvest the saphenous vein by making incisions in the leg. The harvested vein will be grafted into the heart for the bypass surgery. (The saphenous vein is the large,

subcutaneous, superficial vein of the leg and thigh. The vein is often removed by vascular surgeons and used for auto transplantation in coronary artery bypass operations, when arterial grafts are not available or many grafts are required.)

Step02. Chest incision: Make an incision down the centre of the patient's chest, then separate the two halves of the breastbone with a sternal retractor (is a surgical instrument used to separate the two halves of the breastbone by stretching the two halves in opposite direction.)

Step03a. Open membranous sac: Make an incision down the centre of the membranous sac (pericardium- is the thin, sac-like membrane that surrounds the heart. It has two layers: the serous pericardium and the fibrous pericardium) which encloses the heart.

Step03b. Stopping the heart: A clamp is placed on the aorta and a potassium-rich solution is injected to stop the heart.

Step03c. Heart-lung bypass machine: Tubes are attached to the right atrium - sending blood to bypass machine - and the aorta, receiving blood from the bypass machine.

Step04. Graft incision: Make a slit into the affected coronary artery wall at a site below the obstruction to attach the graft. The slit will be enlarged to produce a narrow oval opening.

Step05. Sewing graft onto artery: Carefully sew the lower end of the graft vessel onto the coronary artery using very fine synthetic suture material.

Step06. Bypass machine rewarms blood: Once all the bypass grafts are completed (only one example shown here) the heart-lung bypass machine is used to slowly rewarm the patient's blood.

Step07a. Attach pacemaker wires to heart: Attach thin wires for a temporary pacemaker to the surface of the heart (right & left atrium). A pacemaker corrects any irregular heartbeats which may occur after the bypass machine is stopped. The pacemaker will stay with the patient in intensive care.

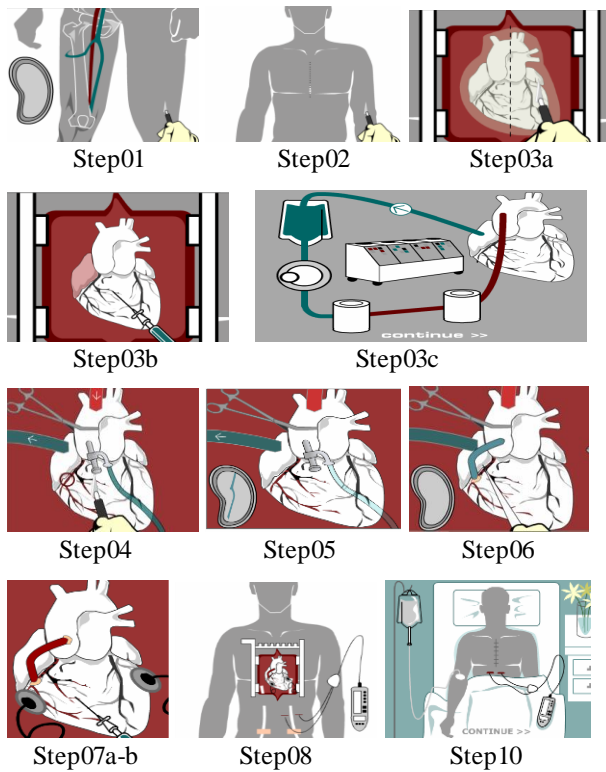


Fig. 2: Steps performed in Heart Bypass Surgery [11]

Step07b. Injection of Protamine: Inject the drug Protamine into the right ventricle. Protamine reverses the Herapin blood-thinning effect by enabling the patient's blood to clot normally again.

Step08. Insert tubes to drain excess fluid: Place tubes into the chest to drain the chest cavity of excess fluid.

Step09a. Closing the breastbone: Stitch up and close the two halves of the breastbone using stainless steel wire.

Step09b. Stitching up the chest: Suture the skin shut to complete the surgery.

Step10. Patient recovery in intensive care: The pacemaker wires and drainage tubes are removed before the patient is discharged.

For our research perspective we will discuss only abstract wording in surgery, for ex, if we talk about Heart bypass Surgery, it is an lengthy procedure but we treat this procedure like an algorithm in few steps for better understanding - [6, 8, 11].

- Step1. Harvesting the saphenous vein.
- Step2. Chest incision.
- Step3a. Open membranous sac.
- Step3b. Potassium-rich solution is injected.
- Step3c. Attach Heart-lung bypass machine.
- Step4. Graft incision.
- Step5. Sewing graft onto artery.
- Step6a. Attach pacemaker machine.
- Step6b. Injection of Protamine.
- Step7. Draining of excess fluid.
- Step8. Closing the breastbone: Stitching using stainless steel wire.
- Step9. Stitching up the chest.

We have identified the primary planning operators corresponding to the above steps.

Step1a:

Incision-Leg(s: scissor-type) scissor-type take on values scissor1, scissor2, ...

PRECOND: s = scissor2

EFFECT: visible (sv)

Saphenous vein is denoted by sv

Step1b:

Harvesting (v: vein, o: object)

PRECOND: v = sv and o = tray and visible (v)

EFFECT: veinIn (v, o)

Step2:

Incision-Chest(s: scissor-type)

PRECOND: $s = \text{scissor1}$

EFFECT: visible (membranous sac)

Membranous sac is also called pericardium.

Step3a:

Incision-Sac(s: scissor-type)

PRECOND: $s = \text{scissor3}$ and visible (membranous sac)

EFFECT: visible (heart)

Step3b:

Stop-heart

PRECOND: heart(running) and visible(heart) and clamp-on-aorta and potassium-soln-injected

EFFECT: $\neg \text{heart(running)}$

Placing a clamp on the aorta is an action. We assume here that the action has been performed. The result of the action is denoted by the truth of the proposition “clamp-on-aorta”. Injecting a potassium rich solution is another necessary action that is to be performed. We also assume here that the action has been performed. The result of the action is denoted by the truth of the proposition “potassium-soln-injected”.

Step3c:

Heart-lung-bypass

PRECOND: $\neg \text{heart(running)}$ and tubes-attached-right-atrium and bypassmachine-connected-aorta

EFFECT: aorta-receiving-blood

When the heart is functioning properly, blood flows from the atrium to the aorta. When the heart is stopped, the flow is maintained by a bypass machine that takes input from the right atrium and pumps blood to the aorta. The machine does both purification and pumping of blood.

Step4:

Graft-incision(affected-coronary-artery-wall: ca)

PRECOND: $\neg \text{heart(running)}$ and slit-at(ca)

EFFECT: graft-inserted(ca)

Step5-6:

Attach-pacemaker-wires

PRECOND: $\neg \text{heart(running)}$ and graft-inserted(CA) and blood-flowing(CA)

EFFECT: pacemaker-wire-attached

The steps 7 to 9 are trivial. The planning operators for these steps are not shown.

III. Modeling the Heart Surgery Domain as a Two Agent System

In tele-surgery, the task of a surgeon (the activities related to a surgery) is partially executed by a robot. Typically, the robot is under the control of a surgeon; it executes the instructions of the controlling surgeon. We model a tele-surgery domain as a two agent system. We call the two agents INTERN and EXPERT. INTERN represents the robot. EXPERT represents the surgeon.

3.1 Transition System

The state space of each agent is a transition system. The states of the transition system are explained below.

1) **States:** Measurements like Body Temperature, Heart Rate, Blood Pressure, Pulse Oxymetry jointly constitute a state.

2) **Actions and Transitions:** An action is specified in terms of the preconditions that must hold before it can be executed and the effects that results when it is executed [1]. Let a be an action with precondition P and effect E . If the state s satisfies P then there is a transition from s to s' due to the action a and s' satisfies E (Fig. 3).

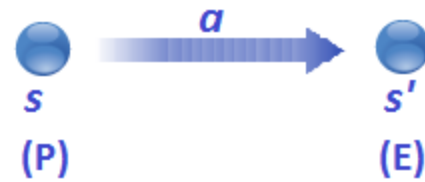


Fig. 3: State Transition

3.2 Agent Communication

INTERN observes and sends the initial state s_0 to the EXPERT. In all subsequent steps, the INTERN on receiving a message (having the information of the next action a) from the EXPERT does the following:

1. Executes the action a
2. Observes the state s' resulting from executing a
3. Send s' to the EXPERT

EXPERT on receiving state s' from the INTERN, it first compares s' with the goal state g ; if both are same then it sends a message to INTERN saying that the surgery is completed. Otherwise, it makes a decision about next action to be performed on s' (Fig. 4).

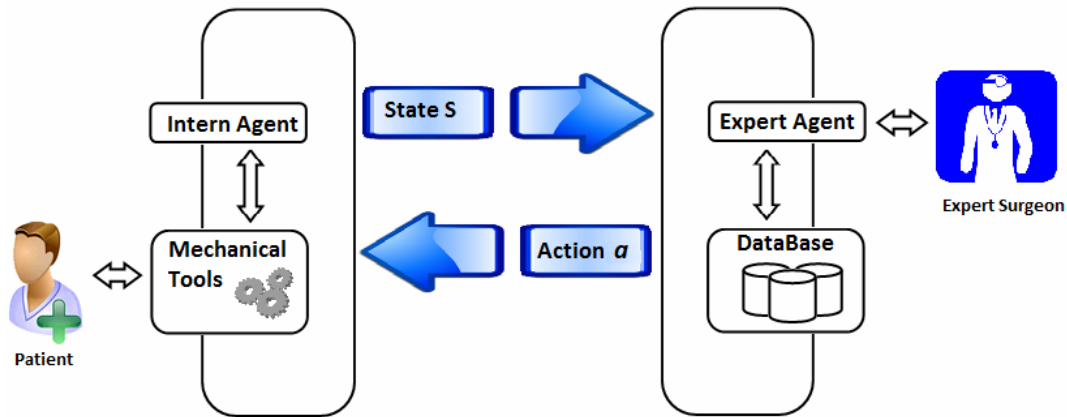


Fig. 4: Communication Process between the EXPERT and INTERN

Decision Making: when EXPERT receives the message it will fetch the “State number” and “state_value” from the message and match the state_value with NORMAL and ABNORMAL set of values (Table I). If the value lies in the range of NORMAL values, it will consult database of surgical procedure in order to get next action, otherwise, it will consult database of recovery procedure in order to get alternate action to recover previous state.

Table 1: Possible values of a State parameter

Possible Values for Body Temp.(°F)	
Normal Values	Abnormal Values
{97 to 99}	{100 to 104}

In communication the message formats for both EXPERT and INTERN will be different. The first field of the message format (sent by the INTERN) will specify the State and the second field is used to store the data belongs to the current state parameters; it may have body temperature, blood pressure, heart rate and many other surgery related data as parameters. In case of message format sent by the EXPERT the first field will consist the message identifier having one of the values among A, R, G and X. Where ‘A’ represents next action to be perform and ‘R’ represent a recovery action, G represents the goal state and X indicates the surgery abortion. The second field tells the details about the action. (Shown in Fig. 5 and 6).

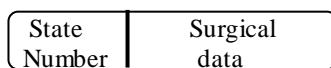


Fig. 5: Message Format for INTERN



Fig. 6: Message Format for EXPERT

3.3 Calculus of Communicating Systems (CCS)

The Calculus of Communicating Systems (CCS) is algebra for specifying and reasoning about concurrent systems [12]. CCS provides a set of terms, operators and axioms that can be used to write and manipulate algebraic expressions. The expressions define the elements of a concurrent system and the manipulations of these expressions reveal how the system behaves.

1) Syntax of CCS [12]

We need to follow two basic principles:

- (a) define atomic processes to model the simplest possible behavior.
- (b) define composition operators to build more complex behavior from simpler ones.

No behavior is the simplest possible behavior. Zero models a system that is either deadlocked or has terminated and Zero is the only atomic process of CCS. When modeling we use a name p to denote an input action and a co-name \bar{p} to denote an output action. We use τ to denote the distinguished internal action.

If S is a process we write $a.S$ to denote the prefixing of S with the action a . $a.S$ models a system that is ready to perform the action, a , and then behave as S .

$$a. S \rightarrow S$$

$S + R$ models a process that can behave either as S or R and $S|R$ models a process in that both behave concurrently. If U is a process constant and V is a process we write

$$U \stackrel{\text{def}}{=} V$$

to give a recursive definition of the behavior of U (recursive if V invokes U).

A labeled transition system(LTS) is a triple -

$$(\text{Proc}, \text{Act}, \{ \xrightarrow{\alpha} \mid \alpha \in \text{Act} \}),$$

Where Proc is a set of processes (states), Act is a set of actions and last set is the transition set between two states or processes. LTS is often beneficial to think of a (finite) LTS as something that can be drawn as a directed (process) graph.

2) CCS regarding Simulation

Agents are *I* and *E* corresponding to intern and expert. Processes corresponding to Intern are I_0 & I_1 and corresponding to Expert are E_0, E_1 and E_2 . These may be considered as intermediate states (or processes). Actions are defined as p : performing action by intern, τ_1 : transferring state data (sending by intern and receiving by expert), α : state verification by expert, β : extracting

next sequential action, τ_2 : transferring action data (sending by expert and receiving by intern).

The Labeled Transition System is used

$$\{ \{I_0, I_1, E_0, E_1, E_2\}, \{p, \alpha, \beta, \tau_1, \tau_2\}, \\ \{I_0 \xrightarrow{\tau} E_0, E_0 \xrightarrow{\alpha} E_1, E_1 \xrightarrow{\beta} E_2, E_2 \xrightarrow{\tau_2} I_1, \\ I_1 \xrightarrow{p} I_0\} \}$$

corresponds to the graph (Figure 7)

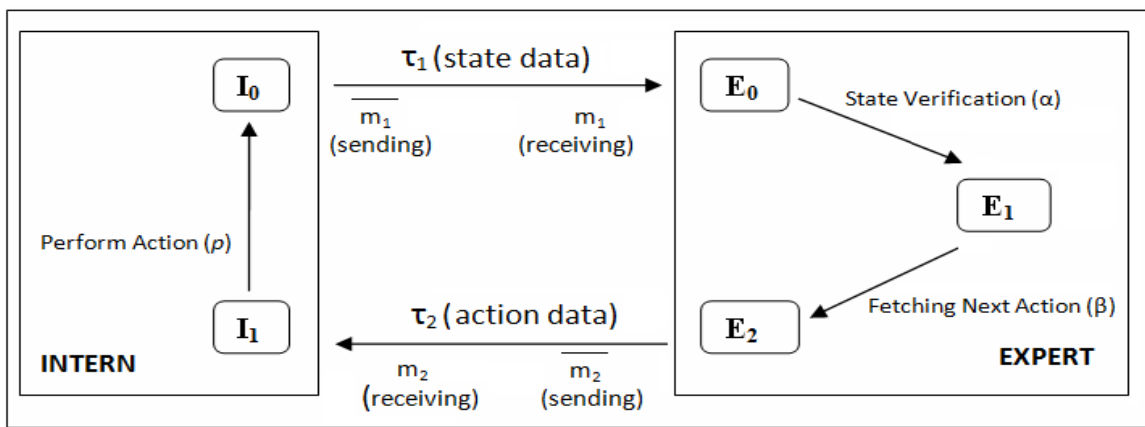


Fig. 7: Communication behavior of the Agents

Behavior:

$$I \stackrel{\text{def}}{=} (\overline{m_1} . m_2 . p . I) \text{ and } E \stackrel{\text{def}}{=} (m_1 . \alpha . \beta . \overline{m_2} . E)$$

$$I | E \text{ or } (\overline{m_1} . m_2 . p . I) | (m_1 . \alpha . \beta . \overline{m_2} . E)$$

$$\xrightarrow{\tau_1} (m_2 . p . I) | (\alpha . \beta . \overline{m_2} . E)$$

$$\xrightarrow{\alpha} (m_2 . p . I) | (\beta . \overline{m_2} . E)$$

$$\xrightarrow{\beta} (m_2 . p . I) | (\overline{m_2} . E)$$

$$\xrightarrow{\tau_2} (p . I) | E$$

$$\xrightarrow{p} I | E \text{ (Recursive)}$$

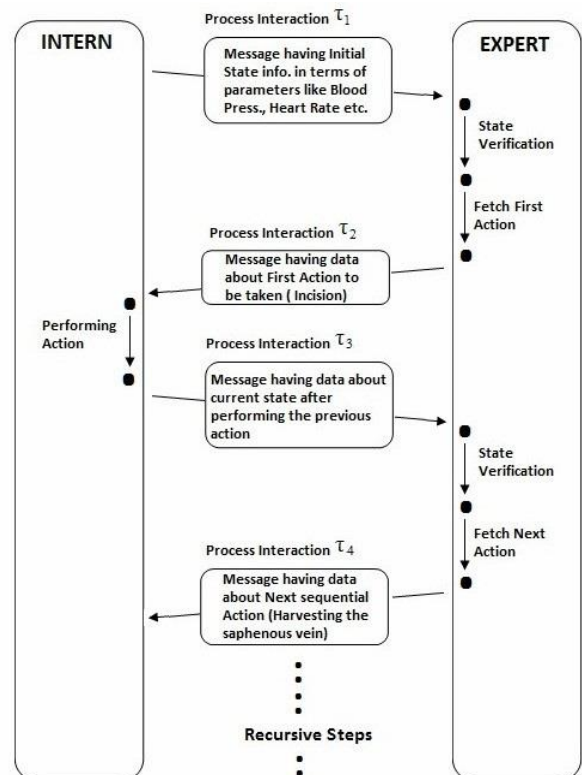


Fig. 8: Intern and Expert interaction

The above behavior shows a recursive procedure of surgery. After performing few steps intern will terminate. Termination may have two reasons here as either the goal is achieved or the surgery is aborted by the expert side. In both the cases message will be sent to intern in order to terminate the surgery. Few steps are shown in the following figure. (Figure 8)

3.4 Multiagent Planning

In the heart surgery domain a joint activity consist of sending an instruction by the surgeon, performing the instruction by the robot upon receipt, and sending the resulting state back to the surgeon. Thus ‘sending an instruction’ and ‘performing the instruction’ would constitute a joint action. We shall from now look at actions from the perspective of the INTERN. In the heart surgery domain a joint activity consist of sending an instruction by the surgeon, performing the instruction by the robot upon receipt, and sending the resulting state back to the surgeon. Thus ‘sending an instruction’ and ‘performing the instruction’ would constitute a joint action. We shall from now look at actions from the perspective of the INTERN.

We denote a joint action as $\langle \text{receive}(a); \text{perform}(a) \rangle$. The meaning of this is that first the INTERN receives an instruction (an action a) from the EXPERT; then it performs the action a . A sequence of such joint actions that transform an initial state to a goal state is defined as a multiagent plan. Formally, let s_0 be an initial state and s_g be a goal state. Let $\sigma_k = \text{receive}(a_k); \text{perform}(a_k)$ where $k \geq 0$. Let $\pi = \sigma_0; \sigma_1; \dots; \sigma_{n-1}$. π is a multiagent plan if it transforms s_0 to s_g . That is $\delta(s_0, \pi) = \delta(\delta(s_0, \sigma_0), \pi') = \delta(\delta(s_0, a_0), \pi') = s_g$, where δ is the transition function of the state transition system of the INTERN and $\pi' = \sigma_1; \dots; \sigma_{n-1}$. It can be easily seen that applying the transition function n times would result in the final state.

Having defined the meaning of a multi-agent plan in the context of the tele-surgery domain, we now develop a game theoretic framework to discuss the existence of such a plan.

Suppose that a state s satisfies all the preconditions of a step of a surgery (call it action a). Under the assumption that the surgery step is actually performed as it is expected to be, then the outcome is known in advance. So the resulting state is unique. The state transition is then said to be deterministic. Now for the expert we can say that almost all the time the resultant state is indeed the desired or expected state. Such a guarantee cannot be associated with the actions performed by the intern. Thus it would be appropriate to model the behavior of the intern as a nondeterministic transition system. Now the expert is aware of some common errors that may be performed inadvertently by the intern. In order to capture this we add for each action of the expert some possible resulting states arising out of such situations. It should be observed that the expert is not actually performing an action. It

merely suggests an action for the intern. From the state resulting from such an action by the intern, the expert identifies whether the state is equivalent to the desired state or a state equivalent to some possible resulting states.

It may so happen that the error performed while executing an action by the intern is so serious that from the resulting state it is very difficult to continue further steps of the surgery. In this case the surgery fails. That is the situation has gone beyond control. The surgery stops after some finite sequence of activities, thereafter no further transition is possible.

A formal representation is given below.

T-expert: denotes the transition system for the expert. (S, Act, s_0, F)

T-intern: denotes the transition system for the intern. (S', Act, s'_0, F')

Let $B \subseteq (S \times S')$ be a binary relation on the states of T-expert and T-intern.

Such a relation is called a bisimulation,

If (p, q) is in B and for all a in Act ,

If (p, a, p') then there exists q' in S' such that (q, a, q') and (p', q') is in B .

Symmetrically, If (p, q) is in B and for all a in Act ,

If (q, a, q') then there exists p' in S such that (p, a, p') and (p', q') is in B .

Let the states of the expert be denoted by p, p', \dots and those of the intern by q, q', \dots

Let the pair $(p, q), (p', q'), \dots$ indicate the equivalent states in the bisimulation B for N-expert and N-intern.

Let the intern execute an action a at q . So the resulting state for the intern is q' . Now the expert selects an action a at p such that the resulting state p' is equivalent to q' .

The game is finite, i.e., it stops after a finite number of moves.

Conditions:

1. If the expert fails to find valid transitions to match the intern's move then the intern wins.
2. The game reaches a state (p, q) from which no further transitions are possible. The expert wins.

Theorem: A surgery is successful iff there is a winning strategy for the expert.

Proof. Let the surgery is successful. This means that a final state is reached and from which no further transition is possible. This as per condition 2 is when the expert wins; that is the expert has a winning strategy for the game.

For the other direction: When the expert has a winning strategy, it means it makes the game come to a state from where no further transition is possible. This in turn means that the surgery is successful.

IV. Implementation Details

In this section we discuss how we have simulated the heart surgery domain based on the model suggested in

section III. We have used two laptops for the two agents. Each laptop has the following configuration:

- 2 GB RAM, 320 GB Hard Disk
- 32 bit OS, 2.20 GHz Processor (Intel i3)

The communication of the agents is synchronous and it was implemented using Java Socket programming. We used AWT and SWING java APIs to design graphical interface (Fig. 9).

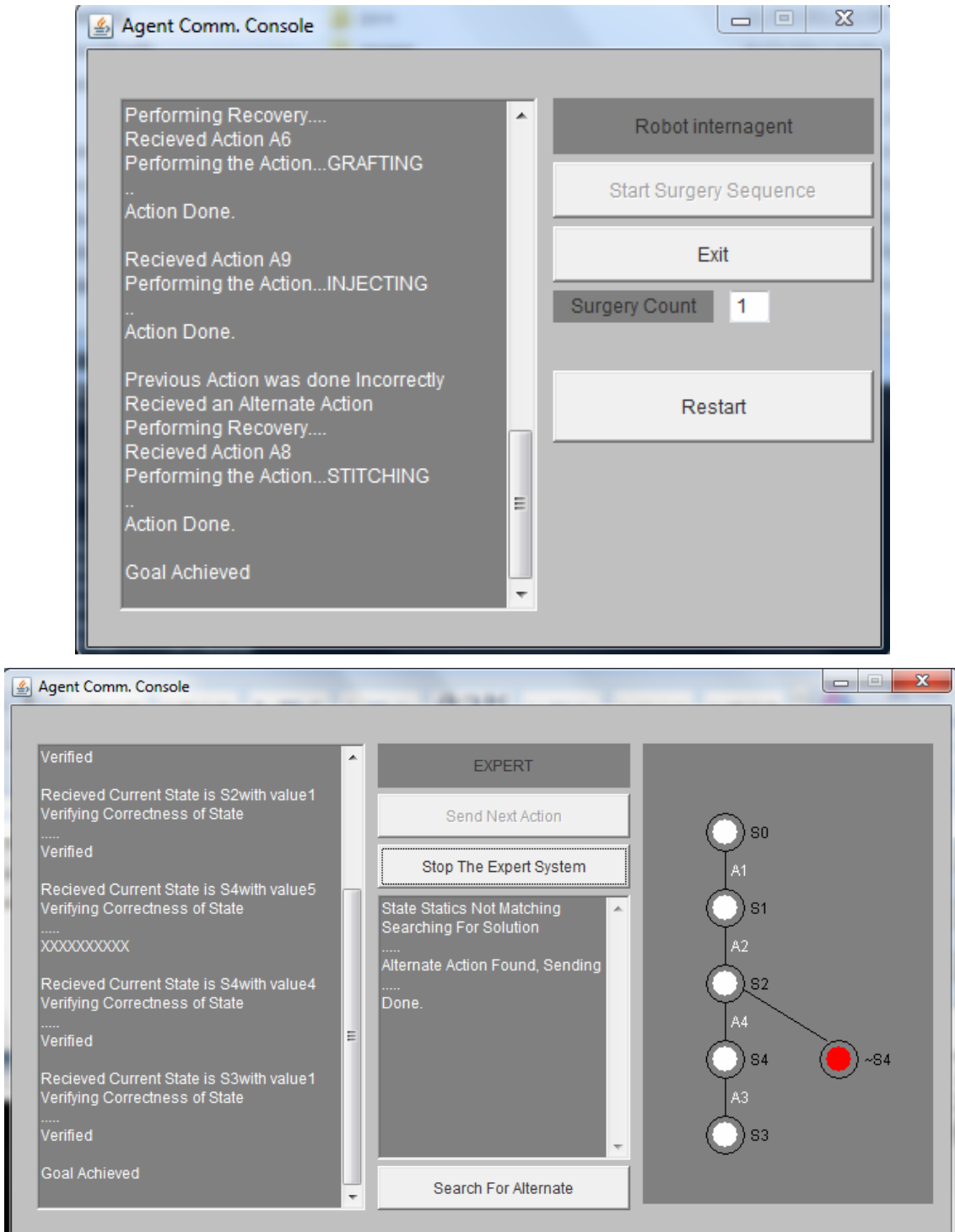


Fig. 9: Snapshots of Tele-surgery Simulator

4.1 System Design

The following block diagrams (Fig. 10 and 11) for INTERN and EXPERT shows how the simulation is designed and how it works. There are two major/main programs running on two different computer systems. Initially the intern main program will interact through messages with the expert main program in order to

transfer the state statistics (blood pressure, body temperature, heart rate etc) of the patient. Now expert will verify the state statistics and accordingly reply to the intern main program. This work is done by a functional module “*verify_state(..)*” which will return a boolean reply as true or false, true indicates the either the patient can go under surgery or the previous action is done well by intern agent.

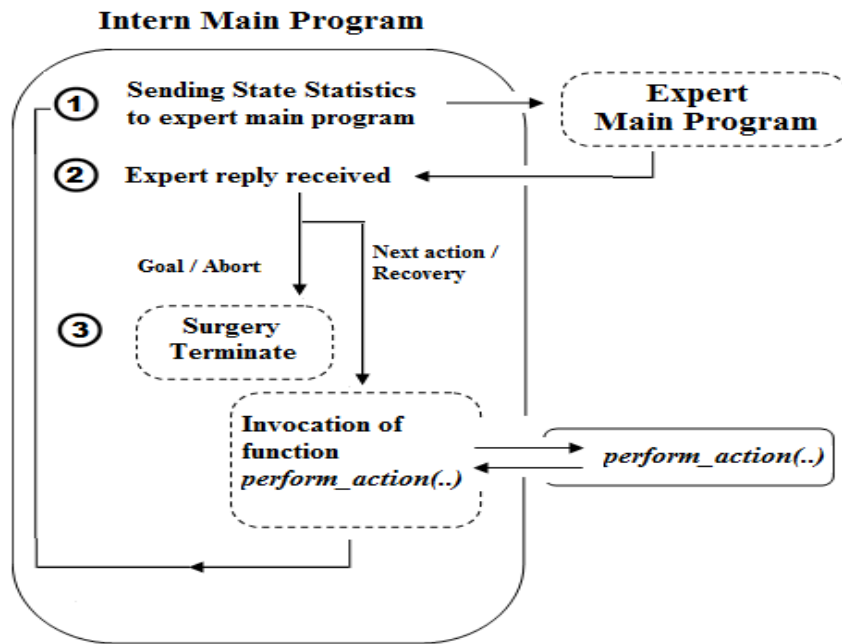


Fig. 10: Block diagram of INTERN indicating different Functional Modules of the System

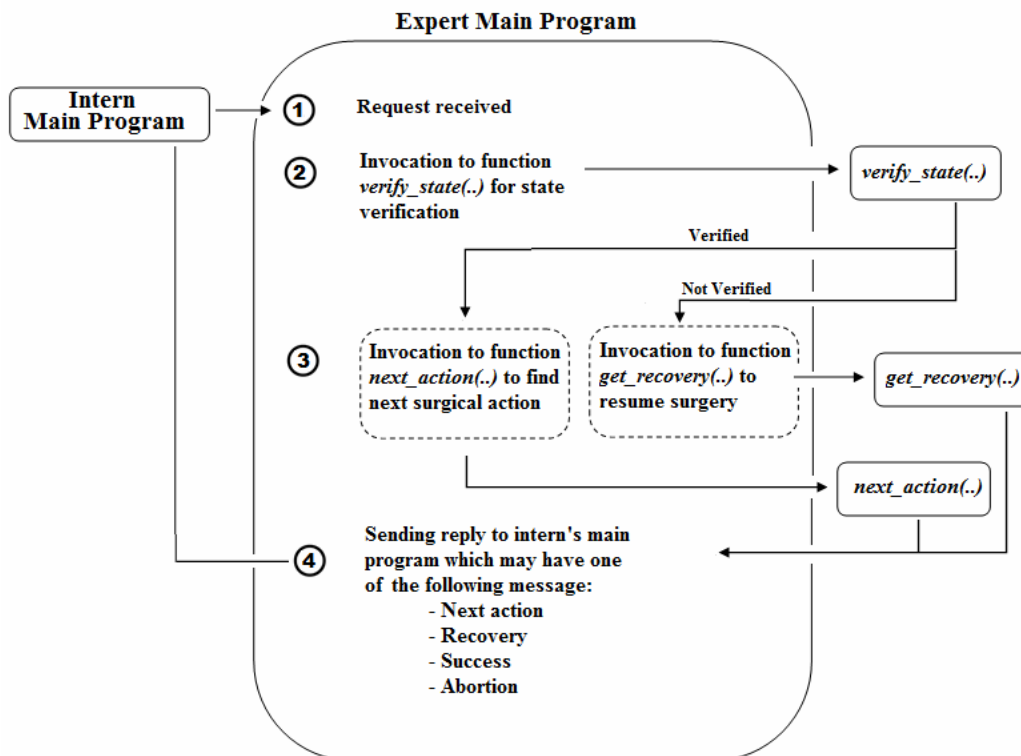


Fig. 11: Block diagram of EXPERT indicating different Functional Modules of the System

In this case it will redirect control to the functional module “*next_action(..)*” which returns the next sequential action. If the functional module “*verify_state(..)*” returns false, it means intern agent have taken an ambiguous action which needs to be correct. In this case control is redirected to functional module “*get_recovery(..)*” which respond by an action of recovery (if recovery possible) or by an action of surgery abortion. The finalized response is sent to the intern main program.

Expert’s response will have one of the following two cases:

- (a) Program will terminate if the response has a message of successful surgery or surgery abortion.
- (b) Program will invoke the functional module “*perform_action(..)*” for action execution and will again send the new state statistics to the expert.

4.2 Functional Modules

perform_action(): This module is responsible for performing the action received from expert main program. This functional module is nothing but a dummy delay function which just makes the feeling of action execution. It also produces the new state statistics from a set of values using a defined probability.

Algo:

1. *Input (new action message)*
2. *Extract the fields’ values.*
3. *Call delay() function. (performing action)*
4. *Generating new parameters values (from the set of normal and abnormal values on the basis of defined probability.)*
5. *Send this new state statistics to EXPERT.*

verify_state(): This module compares the correct statistics with the one received from intern and decides whether the statistics are correct or not, on this basis it invokes either “*next_action()*” or “*get_recovery()*” function.

Algo:

1. *Input (new message having state statistics)*
2. *Extract the parameter values as P_1, P_2, \dots, P_n .*
3. **For** $i=1$ to n

If $NormalV_{min} \leq P_i \leq NormalV_{max}$

then

- if next-action exist

then fetch next action as msg.

else put goal as msg.

else

- if Recovery exist

then fetch recovery as msg.

else put abort as msg.

next_action(): This function maintains the sequence of actions for a particular surgery through a data table. This table stores the mapping of states and actions in a well defined order and treated as blue print for surgery. This function takes previous state and action as inputs to fetch next sequential action.

Algo:

1. *Input old state number.*
2. *Connect database.*
3. *Search the next action in the procedural database table using the state number as the keyword.*
4. *if next-action exist*

then fetch next-action and return it.

else return goal.

get_recovery(): The control is redirected to this functional module only when the state is found as ambiguous. Action can be done incorrectly in many manners and each false state may have many recovery options thus it is somewhat complex. But we have used an easier version of this function by reducing some of its capability by providing only one recovery solution for each mistake.

Algo:

1. *Input ambiguous state number.*
2. *Connect with the database.*
3. *Search the recover action in the recovery matrix table using the ambiguous state number as the keyword.*
4. *If the corresponding row of ambiguous state number has any 1’s in matrix, it means recovery possible and the appropriate recovery is sent to the INTERN. If there is no 1’s in the row which means recovery is not possible and need to abort the message.*

main program : The main program of intern is simple one that just perform 3 steps recursively until it finds its goal or abort message from expert main program. Expert’s main program keeps track of the previous states and actions.

4.3 Examination of States

When the INTERN sends a state, the EXPERT on receiving the message tries to find out which action is

applicable at this state. For this the EXPERT examines the surgical data field in the message. For instance if temperature is a parameter whose value is chosen as say 99, by a random generation process. The EXPERT checks whether the value is within the desirable limits. If yes, it sends the suitable action. Otherwise it will reply saying that the state is not satisfying the expected criteria.

4.4 Recovery in Case of Surgical Mistakes

Performing an action in a medical surgical domain may not always result in the appropriate or desirable state. This may be due to several factors, like mistakes committed inadvertently and improper functioning of an instrument. We have implemented one such simple scenario for the heart surgery domain. We refer to Figure-12 in the Data

Current State	Next Action	Next State
S0	A1	S1
S1	A2	S2
S2	A4	S4
S4	A5	S5
S5	A3	S3
S3	--	--

	S0	S1	S2	S3	S4	S5
~S0	0	0	0	0	0	0
~S1	0	1	1	0	0	0
~S2	0	0	0	0	0	0
~S3	0	0	0	0	0	0
~S4	0	0	0	1	1	1
~S5	0	0	0	0	0	0

S0: Initial State, S3: Goal State

Fig. 12: Data Structure for Surgical Data storage

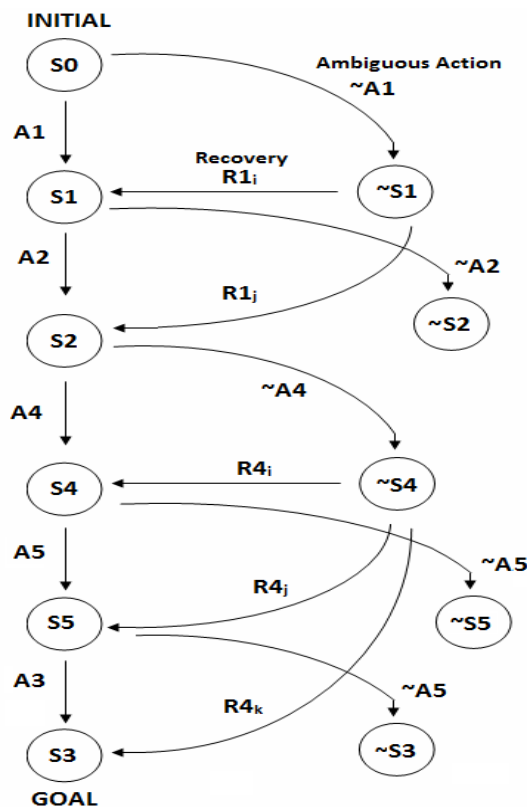


Fig. 13: Transition System in Surgery

Table given in the left hand side, performing A1 at S0 results in S1. However due to some mistakes the resulting state is not S1; we denote this new state by ~S1. There are two recovery actions at ~S1 namely R1i and R1j as shown in Figure 13. The resulting states are S1 and S2 respectively. In the first case we get back to the original desirable state S1. In the 2nd case we go to S2. Sometimes recovery is not possible as in ~S2.

V. Conclusion

In this paper we have developed a formal model of a tele-surgery domain. For this study we have chosen the heart surgery domain. We defined the steps of the surgery as planning operators. We have shown that performing a tele-surgery amounts to finding a multiagent (here two-agent) plan that involves joint actions. We have obtained a condition for successful surgery. The computer simulations reflect the complexity of the task at hand. We have also implemented the real life features for the surgery domain where errors may occur inadvertently. The formal model presented here can easily be applied for other surgery domains as well. We expect to develop simulations for other surgery domains as well as part of our future work.

Acknowledgments

The second author acknowledges the support of the DST grant SR/S3/EECE/0075/2011 (G).

References

- [1] S. Russell and P. Norvig, "Artificial Intelligence: A modern Approach", Prentice Hall, 1995
- [2] "Remote Surgery", http://en.wikipedia.org/wiki/Remote_surgery, 2012.
- [3] K. Brady and T. J. Tam, "Internet-Based Remote Teleoperation", Proceedings of the 1998 IEEE Int. Conf. on Robotics and Automation, 1998.
- [4] A. Ferworm, R. Roque, I. Vecchia and MAX, "Wireless teleoperation via the World Wide Web", Proc. IASTED Conf. on Robotics & Applications, 1999.
- [5] H. Hu, L. Yu, P. W. Tsui, Q. Zhou, "Internet-based Robotic Systems for Teleoperation", International Journal of Assembly Automation, Vol. 21, No. 2, 2001.
- [6] "The Circulatory System" <http://lsa.colorado.edu/essence/texts/heart.html>
- [7] A. Moreno, "Medical Applications of Multi-Agent Systems", Computer Science & Mathematics

Department, Universitat Rovira i Virgili, Spain, 2003.

- [8] "Coronary Artery Disease", Surgical Associates of Texas, P.A., Texas heart Institute, April 2005.
- [9] M. de Weerd, A. ter Mors, and C. Witteveen. "Multi-agent planning-an introduction to planning and coordination". Technical report, Delft University of Technology, 2005.
- [10] J. J. Regan, MD and E. C. Benzel, MD, "Robotics and Computers in Minimally Invasive Spine Surgery, January 2nd 2010.
- [11] "Virtual Open Heart Surgery", <http://www.abc.net.au/science/lcs/swf/heart.swf>
- [12] A.K. Lal and R. Niyogi, "Formal Modeling of a Tele-surgery domain as a Multiagent planning problem", Third International Conference on Advanced Computing & Communication Technologies, pg 55-58, April 6-7, 2013, Rohtak, India.

Authors' Profiles



Amod Kumar Lal is an M.Tech. student of Computer Science and Engineering at IIT Roorkee. His research interests include communication in multi-agent systems and distributed systems.



Dr. Rajdeep Niyogi is an Associate Professor in the Indian Institute of Technology (IIT) Roorkee. He did his Ph.D. in Computer Science and Engineering from IIT Kharagpur. His research interests include Automated Planning, Formal Methods and Distributed Systems. He is a member

of ACM.

How to cite this paper: Amod Kumar Lal, Rajdeep Niyogi, "A Multiagent Planning Approach to Model a Tele-Surgery Domain", International Journal of Intelligent Systems and Applications(IJISA), vol.5, no.9, pp.27-38, 2013. DOI: 10.5815/ijisa.2013.09.04