

Robotics


Planning (block world)

What is planning in general?

Layout plan that realize task

- **Task**

Given a task → sequence of actions that will carry out.

Assembling  *it is a plan of how to do it*

This is the paper that mention → in d2l

AND/OR Graph Representation of Assembly Plans

LUIZ S. HOMEM DE MELLO, MEMBER, IEEE, AND ARTHUR C. SANDERSON, SENIOR MEMBER, IEEE

Abstract—A compact representation of all possible assembly plans of a given product using AND/OR graphs is presented. Such a representation forms the basis for efficient planning algorithms that enable an increase in assembly system flexibility by allowing an intelligent robot to pick a course of action according to instantaneous conditions. The AND/OR graph is equivalent to a state transition graph but requires fewer nodes and simplifies the search for feasible plans. Three applications are discussed: the preselection of the best assembly plan, the recovery from execution errors, and the opportunistic scheduling of tasks. The same approach used to preselect the best assembly plan can be used to preselect the best disassembly plan. A change in the definition of goal nodes allows the preselection of the best disassembly plan that will release a given part for maintenance or repair tasks. An example of an assembly with four parts illustrates the use of the AND/OR graph representation to find the best assembly plan based on the weighing of operations according to complexity of manipulation and stability of subassemblies. In practice, a generic search algorithm, such as the AO*, may be used to find this plan. A hypothetical error situation in the assembly of the four-part assembly is discussed to show how a bottom-up search of the AND/OR graph leads to an efficient recovery. The scheduling efficiency using this representation is compared with fixed sequence and precedence graph representations. The AND/OR graph consistently reduces the average number of operations.

This paper presents a compact representation for the set of all possible assembly plans for a given product using AND/OR graphs. Such a representation enables an increase in assembly flexibility by allowing an intelligent robot to pick the more convenient course of action according to instantaneous conditions. Section II discusses background to assembly planning. Section III presents the AND/OR graph representation, and Section IV describes its use for the assembly of a simple product. Three applications are discussed. Section V shows how the selection of the best assembly plans can be implemented as a top-down search of the AND/OR graph; it also shows how the same approach can be used to select the best disassembly plan to release a given part for repair. Section VI shows the use of a bottom-up search of the AND/OR graph for recovery from execution errors. Section VII illustrates the use of the representation in opportunistic scheduling. Section VIII summarizes the contribution of the paper and points to further research.

- *Elementary task or elementary operation that composed in a sequence to put together*
- *Time constrains associated with that? When supposed to do this?*

Manufacturing → *resource allocation (do certain operation at certain machines) and scheduling (time in scheduling)*

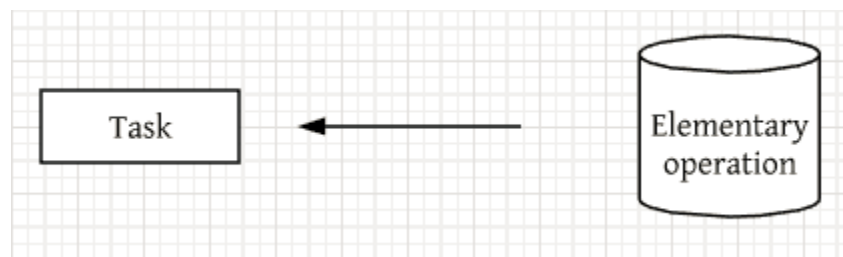
Real time run operating system!

Abstract level Sequences operation and have to sequence in particular way but working with real time system, having time constrain, time requirement to execute certain operations.

- Not dealing with time but dealing with finding sequence of operation that carry out particular

Assume

Having the Task



The abstraction that working with

Focus on very high-level description what robot can do

- Move hand empty from one location to another
- Pick the object
- Carry object
- Holding object
- Place object

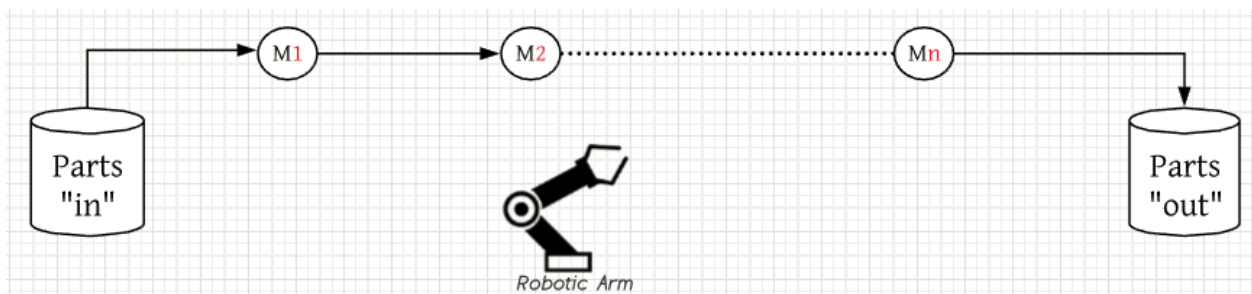
In that level what have is

- Abstract high level "commands"
 - Plan
- path planning → Collision free trajectory planning → fine motion planning

Example

From manufacturing, Parts move from $M_i \rightarrow M_{i+1}$

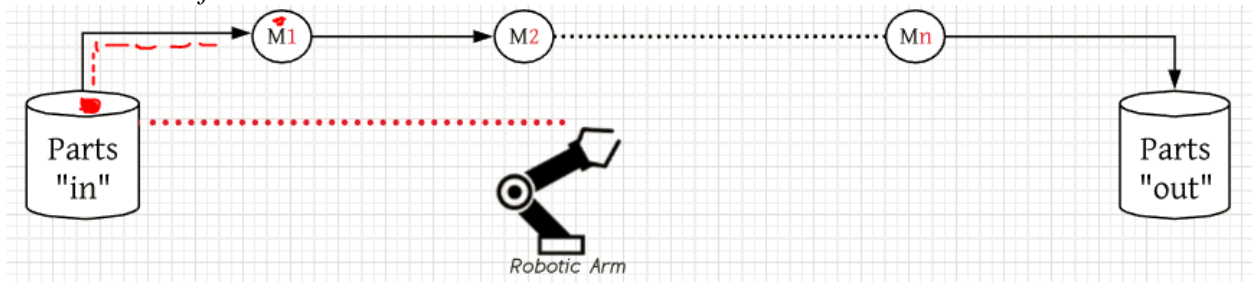
Robot can move to any M_i , parts are moved only by the Robot



How to model these processes?

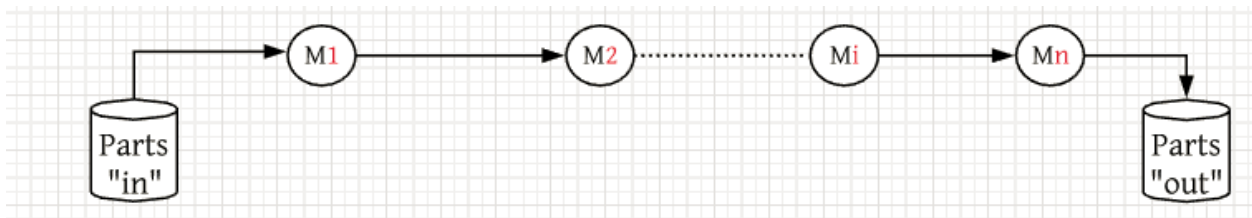
Modeling this process

State of M_i



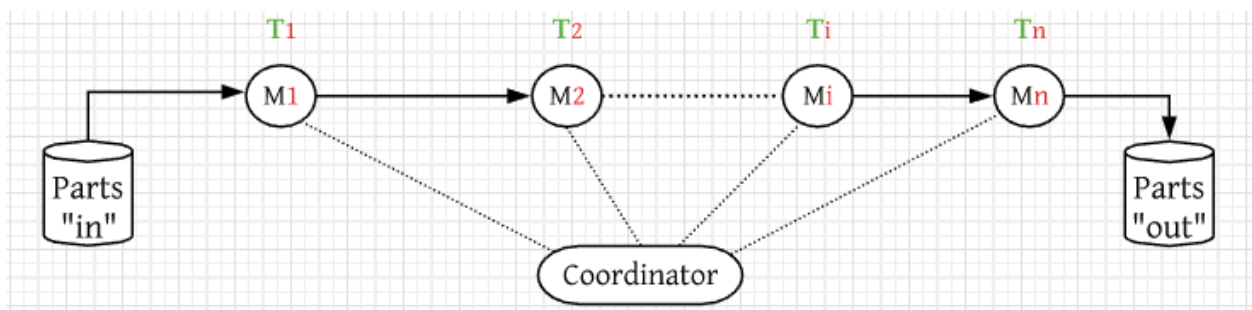
Note

- $M \rightarrow$ machine is separated from the Robot
- Machine does not have memory or buffer
- Machine accepts one part at time
- M_i can be



Vectors $\langle SM_1, SM_2, \dots, SM_n \rangle$ contain all states

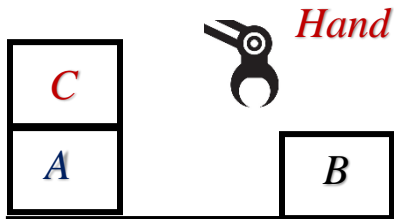
Control strategy ?



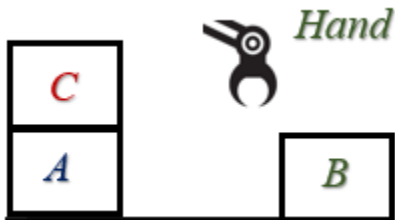
Blocks world → Abstraction

- Use predicate logic to describe the “world” state and robot actions

Example

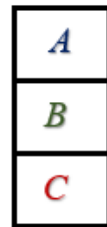


The initial state



The goal state

$[On(B, C) \wedge On(A, B)]$



Define predicate described the state

- $Clear(B)$
- $Clear\odot$
- $On(C, A)$
- $On-table(A)$
- $On-table(B)$
- $Hand\ empty\ (HE)$

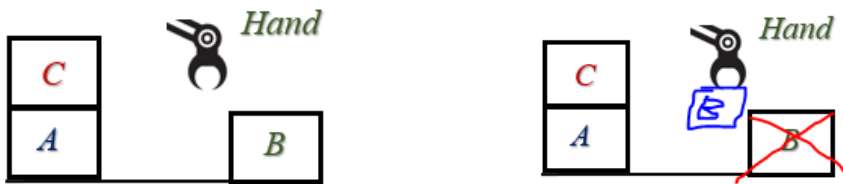
STRIPS

Rules for the Robot (F-Rule)

- Structure rule
 1. Precondition list → State has to match the predicate this list
 2. Delete list → Predicate deleted as result of applying the rule
 3. Add list → Predicate add as a result

Rule

- *Pick up (x)*
 - *Precondition* *On-table(x), HE, Clear(x)*
 - *Delete* *On-table(x), HE, Clear(x)*
 - *Add* *On-table(x)*
- *Substitute < X, B >*
 - *Pick up (B)*
 - *Precondition* *On-table(B), HE, Clear(B)*
 - *Delete* *On-table(B), HE, Clear(B)*
 - *Add* *On-table(B)*



What left from the initial state

New state is \rightarrow *On-table(A), Holding (B)*
On(C,A), clear(C)

Planning in the block world

Artificial Intelligent Production System “AIPS”

- States represented as collection of predicates that describe the situation.
- Initial state
- Goal state

Hand out (in d2l)

7.2. A FORWARD PRODUCTION SYSTEM

The simplest type of robot problem-solving system is a production system that uses the state description as the global database and the rules modeling robot actions as F-rules. In such a system, we select applicable F-rules to apply until we produce a state description that matches the goal expression. Let us examine how such a system might operate in a concrete example.

Consider the F-rules given below, in STRIPS-form, corresponding to a set of actions for the robot of Figure 7.1.

- 1) **pickup(x)**
P & D: *ONTABLE*(x), *CLEAR*(x), *HANDEEMPTY*
A: *HOLDING*(x)
- 2) **putdown(x)**
P & D: *HOLDING*(x)
A: *ONTABLE*(x), *CLEAR*(x), *HANDEEMPTY*
- 3) **stack(x, y)**
P & D: *HOLDING*(x), *CLEAR*(y)
A: *HANDEEMPTY*, *ON*(x, y), *CLEAR*(x)
- 4) **unstack(x, y)**
P & D: *HANDEEMPTY*, *CLEAR*(x), *ON*(x, y)
A: *HOLDING*(x), *CLEAR*(y)

- Focus in the four fundamental rules
 - a) How to generate plan
 - b) What is the good presentation for it in the context of dynamically execute certain actions and it is little different from Artificial Intelligent Production System “AIPS” because **did not do re-planning but did backtracking**
 - c) Search to get optimal planning forward chain and back forward.

7.2. A FORWARD PRODUCTION SYSTEM

The simplest type of robot problem-solving system is a production system that uses the state description as the global database and the rules modeling robot actions as F-rules. In such a system, we select applicable F-rules to apply until we produce a state description that matches the goal expression. Let us examine how such a system might operate in a concrete example.

Consider the F-rules given below, in STRIPS-form, corresponding to a set of actions for the robot of Figure 7.1.

- 1) **pickup**(*x*)
P & D: *ONTABLE*(*x*), *CLEAR*(*x*), *HANDEEMPTY*
A: *HOLDING*(*x*)
- 2) **putdown**(*x*)
P & D: *HOLDING*(*x*)
A: *ONTABLE*(*x*), *CLEAR*(*x*), *HANDEEMPTY*
- 3) **stack**(*x*,*y*)
P & D: *HOLDING*(*x*), *CLEAR*(*y*)
A: *HANDEEMPTY*, *ON*(*x*,*y*), *CLEAR*(*x*)
- 4) **unstack**(*x*,*y*)
P & D: *HANDEEMPTY*, *CLEAR*(*x*), *ON*(*x*,*y*)
A: *HOLDING*(*x*), *CLEAR*(*y*)

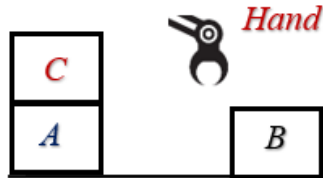
Note that in each of these rules, the precondition formula (expressed as a list of literals) and the delete list happen to be identical. The first rule is the same as the rule that we used as an example in the last section. The others are models of actions for putting down, stacking, and unstacking blocks.

Suppose our goal is the state shown in Figure 7.2. Working forward from the initial state description shown in Figure 7.1, we see that **pickup**(*B*) and **unstack**(*C*,*A*) are the only applicable F-rules. Figure 7.3 shows the complete state-space for this problem, with a solution path indicated by the dark branches. The initial state description is labeled

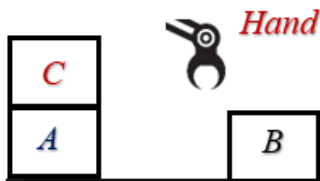
F-Rules

- Pick up
 - Putdown
 - Stack
 - Unstack
- What is stack and unstack?

Example



The initial state



The goal state

$[On(B, C) \wedge On(A, B)]$



States predicates

- Clear(B), On(C, A), On-table(A)
- Clear(C), Hand empty(HE), On-table(B)

Generate plan

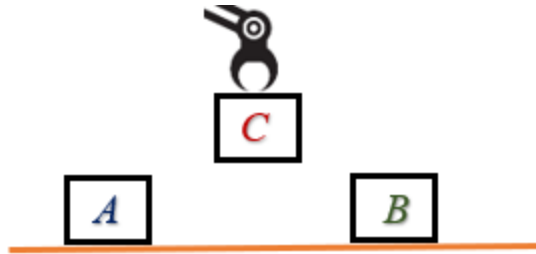
What is the plan?

Sequences of Robot actions from the vocabulary or from the database.

The Plan

4. Unstack(C, A)
5. Putdown(C)
6. Pickup(B)
7. Stack(B, C)
8. Pickup(A)
9. Stack(A, B)

This is the plan



- *Clear(B), On-table(A)*
- *Clear(A), Holding(C), On-table(B)*



- *Clear(B), On-table(B)*
- *Clear(A), On-table(A)*
- *Clear(C), On-table(C)*
- *Hand empty (HE)*



- *Clear(A), On-table(A)*
- *Clear(C), On-table(C), Holding(B)*



- *Clear(A), On-table(A)*
- *On(B,C), On-table(C)*
- *Hand empty (HE)*



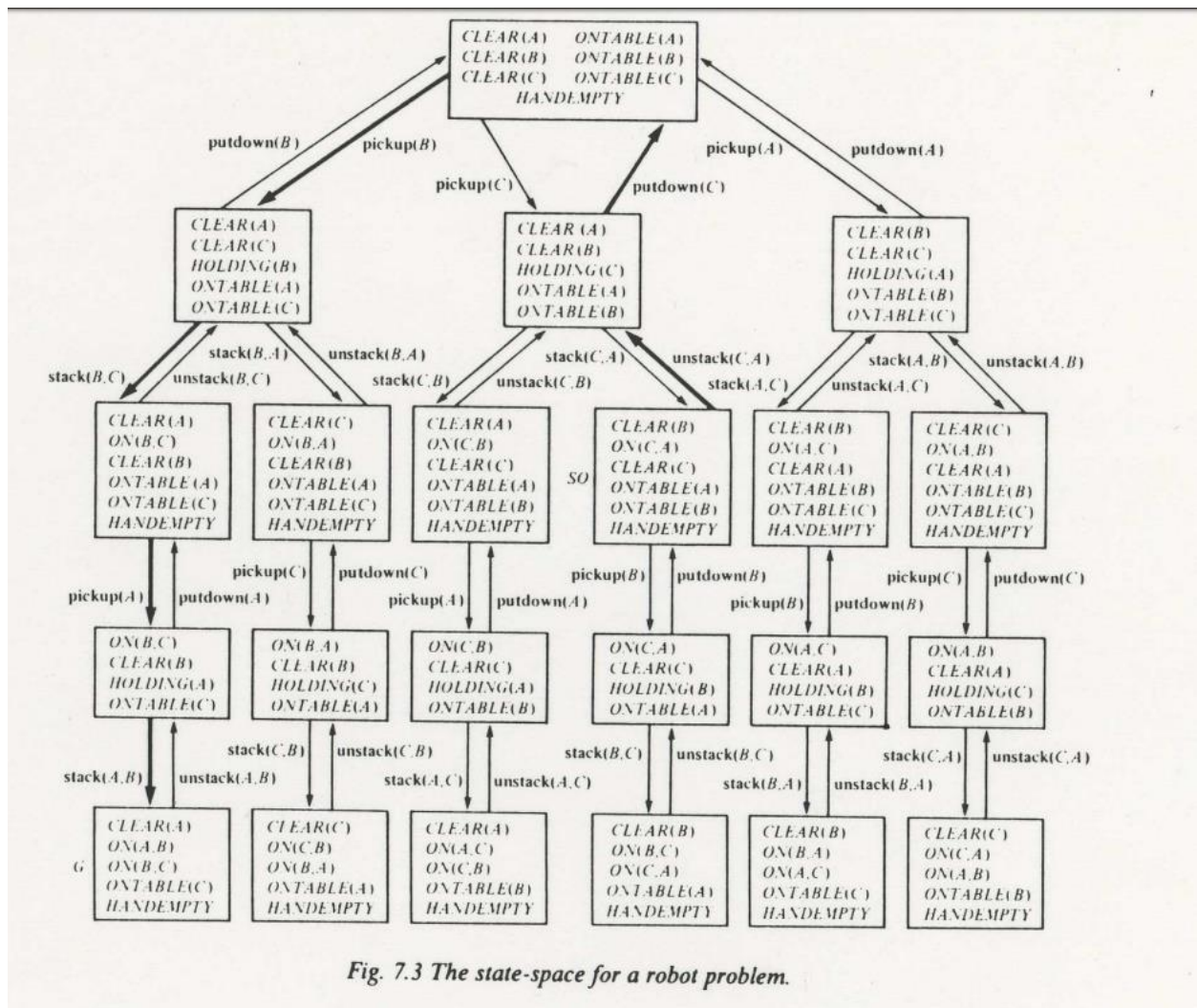
- *Holding (A)*
- *On(B,C) , On-table(C) ,*



- *On (A,B)*
- *On(B,C) , On-table(C)*

Sequences of moves that will be accomplished a particular task

- *Is it online or offline planning?*
- *It is called offline planning (have problem at hand and try to generate plan to accomplish a particular task)*
- *GPS can be both offline and online planning*



- What is the control strategy?
It is graph search
- It can do by forward chaining and backward chaining

Representation for plan (STRIPS planning)

- Triangle table

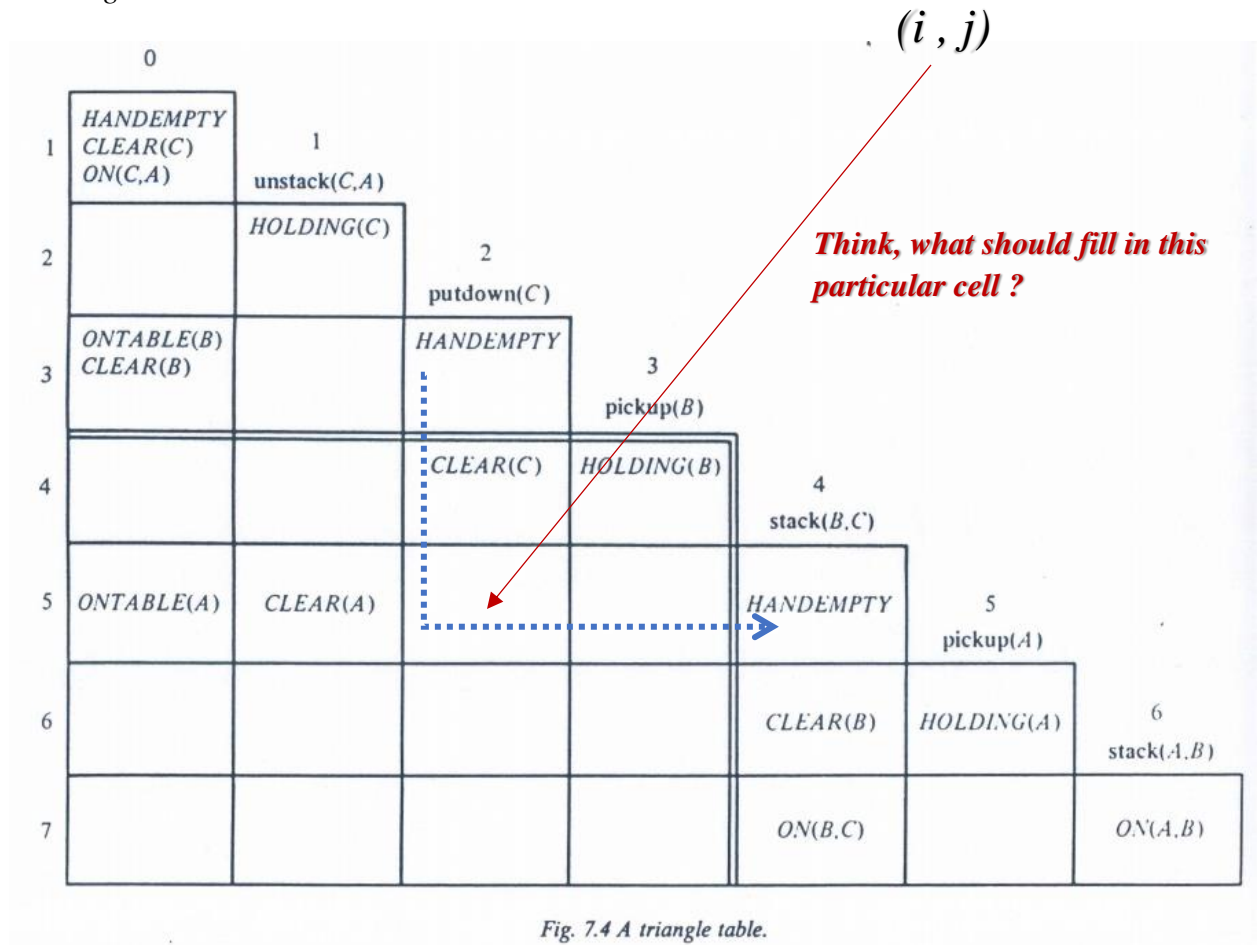


Fig. 7.4 A triangle table.

- What this table contain and what should do!
 - First column place predicates of the initial state description that serve as precondition for rule of plan.
 - Last row, predicate of goal state that stem from the application of the rule

State description

- Clear (B), On (C, A), On-table (A)
- Clear (C), Hand empty (HE), On-table(B)

