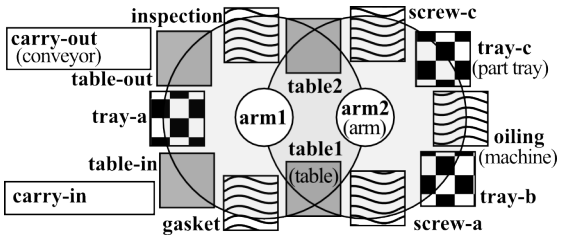
Fully Automated Cyclic Planning for Large-Scale Manufacturing Domains

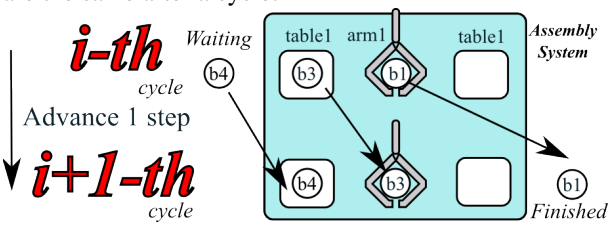
Masataro Asai and Alex Fukunaga
The University of Tokyo

Mass-Manufacturing Domain and Loops



- CELL-ASSEMBLY = Gripper + Woodworking + Logistics
  - Assemble and paint products while moving them with arms between tables or special purpose machines
  - also a *temporal* problem (actions run in parallel)

The **Goal**: solving problems with **OVER 1000 identical products.** For that we used **Loops.** Everythings but indices are the same after a cycle.



Each Loop path is identified by Steady State (SS) ≡ a state in the beginning of one cycle, indexed with i.

ex. S_i : (at p_{i+1} table), (at p_i painter), (painted p_i).

~~However, two difficulties remain.~~

Which states are Steady States?

- Search all states from Init and test them → impractical.
- Labelling some objects (`table`, `machine`) and predicates (`at`) ? → Possible, but domain-dependent

Of all steady states, which one yields the best cyclic plan?

- Brute force search → impractical
- ∵ testing each SS invokes a *standard STRIPS planner each time*

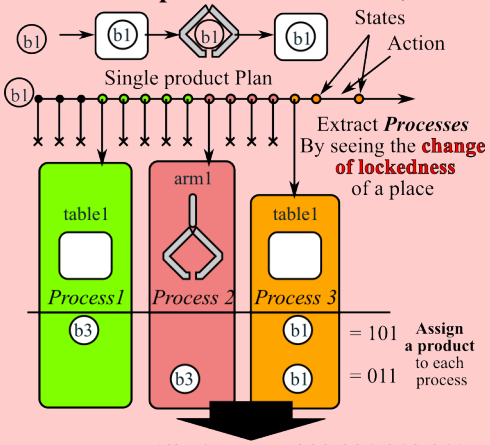
Domain Independent Analysys

Main contribution: Owner / Lock predicates detection for STRIPS

- $o = (P\ X_1\ X_2 ...)$ a place?
- find a *lock* predicate l that satisfies some condition with o. If l exists, o is a place. Condition : for all action a,

1. If a occupies a place, a should check if the place is not in use, and a should acquire the lock.
2. If a leaves the place, release the lock.

Based on the owner/lock predicates, **extract processes from a plan** to enumerate SSs,



Enumerate All Bit vector 000,001,010,011...
=Enumerate All SSs

Pruning SSs

Now we not only enumerate SSs but also *prune* them when:

	Table1	Arm	Table1
Infeasible	1	0	1
Deadlock	1	1	0
Duplicated*	1	0	0
	0	1	0

(* They are the same loop)

Plan each cycle from S_i to S_{i+1}

Get the least-makespan cyclic plan

Generate N instance plan by unrolling i. With no additional computation cost!

A unit capacity resource ≈ a place.

- use table1 ∈ Attatch @ table1.

Place or not? **Easy** for human, **Difficult** for machines. Syntactically they are the same (pred ?p X).

predicate	place? (for human)
(color ?p red)	no,~~maybe~~
(at ?p table1)	yes,maybe

Results Compared ACP vs 5 planners on 5 CELL-ASSEMBLY and modified IPC Temporal Woodworking, unmodified IPC Barman

Problem	♯ of products	run-time	ACP makespan	FD/LM$_{cut}$ scheduler	FD/LAMA scheduler		yahsp	DAE	CPT4
	N	[sec]	c_{ACP}						
CA1	4	1048	**331**	fail	892	807	*774*	fail	
	16	1049	**1255**	fail	fail	fail	fail	fail	
	1024	1050	**78871**	fail	fail	fail	fail	fail	
CA2	4	34	**246**	*249*	256	607	332	fail	
	16	33	**978**	fail	fail	fail	fail	fail	
	1024	35	**62466**	fail	fail	fail	fail	fail	
CA3	4	1893	**660**	fail	*1080*	fail	fail	fail	
	16	1953	**2352**	fail	fail	fail	fail	fail	
	1024	1973	**144480**	fail	fail	fail	fail	fail	
CA4	4	1163	**318**	fail	*540*	715	fail	fail	
	16	1162	**1074**	fail	fail	fail	fail	fail	
	1024	1165	**64578**	fail	fail	fail	fail	fail	
CA5	4	1968	**804**	fail	*947*	fail	fail	fail	
	16	1856	**2508**	fail	fail	fail	fail	fail	
	1024	1894	**145644**	fail	fail	fail	fail	fail	
Woodworking	4	11	**80**	**80**	**80**	150	**80**	**80**	
	16	11	**260**	**260**	330	590	270	fail	
	1024	15	**15380**	fail	fail	fail	fail	fail	
Barman	4	331	35	fail	**23**	81	31	fail	
	16	332	179	fail	fail	fail	fail	fail	
	1024	332	12275	fail	fail	fail	fail	fail	