Reinforced Encoding for Planning as SAT

Tomáš Balyo
Roman Barták
Otakar Trunda

Charles University in Prague

Planning

- Input:
  - Initial state, goal states, available actions
- Output:
  - A sequence of actions that transforms initial state to goal state
- Classical planning:
  - Deterministic, fully observable, static world
  - Actions are instantaneous
  - Domain-independent techniques

SAS+ format

- Initial state:

	1	2	3	4	5	6
A	7	0	2	3	5	5
B	4	2	3	1	0	4
C	3	7	2	3	3	1
D	25	2	1	3	5	0

SAS+ format

- Actions:
  - (A3 = 7, B1 = 4) => (A3 = 1)
  - (D3 = 2, D1 = 6) => (C5 = 1, B2 = 1)
  - () => (C1 = 1)
  - …

- Goal condition:
  - D6 = 1

SAS+ format

- Current state:

	1	2	3	4	5	6
A	7	0	2	3	5	5
B	4	2	3	1	0	4
C	3	7	2	3	3	1
D	25	2	1	3	5	0

- Action: (B4 = 1, C2 = 7) => C5 = 4

SAS+ format

- Current state:

	1	2	3	4	5	6
A	7	0	2	3	5	5
B	4	2	3	1	0	4
C	3	7	2	3	3	1
D	25	2	1	3	5	0

- Action: (B4 = 1, C2 = 7) => C5 = 4

SAS+ format

- New state:

	1	2	3	4	5	6
A	7	0	2	3	5	5
B	4	2	3	1	0	4
C	3	7	2	3	4	1
D	25	2	1	3	5	0

- Action: (B4 = 1, C2 = 7) => C5 = 4

Boolean satisfiability (SAT)

- Input: boolean formula in CNF
- Output: satisfying assignment to variables

  OR

  „*NO*“ if no such assignment exists
- NP-complete problem
- Lots of SAT solvers exist
  - Often effective on practical problems

Planning as SAT

- Solving the planning problem using a SAT solver
  - Popular and competitive approach
- Basic idea:
  - For a planning problem *P* and a number *k*, we create a boolean formula *F*, such that
    - *F* is satisfiable if and only if there is a plan for *P* that contains *k* actions (steps)
    - A plan for *P* of a length *k* can by constructed from a satisficing assingnment to *F*
  - We increase *k* until the formula is satisfiable

Parallel plans

- Some actions can be executed simultaneously
- Parallel steps:
  - Actions *u* and *v* can be in the same parellel step if
    - Effects of *u* don‘t violate preconditions of *v* and vice versa
  - „For all“ – semantics:
    - Set of actions can be in the same parallel step if *all* orderings of the actions form a valid plan
  - „Exists“ – semantics:
    - Set of actions can be in the same parallel step if *there is* an ordering of the actions that forms a valid plan
    - No longer a „parallel“ semantics

Parallel plans

- Some actions can be executed simultaneously
- Parallel steps:
  - Actions *u* and *v* can be in the same parellel step if
    - Effects of *u* don‘t violate preconditions of *v* and vice versa
  - **„For all“ – semantics:**
    - Set of actions can be in the same parallel step if *all* orderings of the actions form a valid plan
  - „Exists“ – semantics:
    - Set of actions can be in the same parallel step if *there is* an ordering of the actions that forms a valid plan
    - No longer a „parallel“ semantics

Parallel plans

- Some actions can be executed simultaneously
- Parallel steps:
  - Actions *u* and *v* can be in the same parellel step if
    - Effects of *u* don‘t violate preconditions of *v* and vice versa
  - Increasing planning efficiency
    - Shorter makespan, less SAT solver calls
    - How do we find actions that can be executed together?
  - Sufficient condition:
    - actions are pairwise **independent** (don‘t share variables)

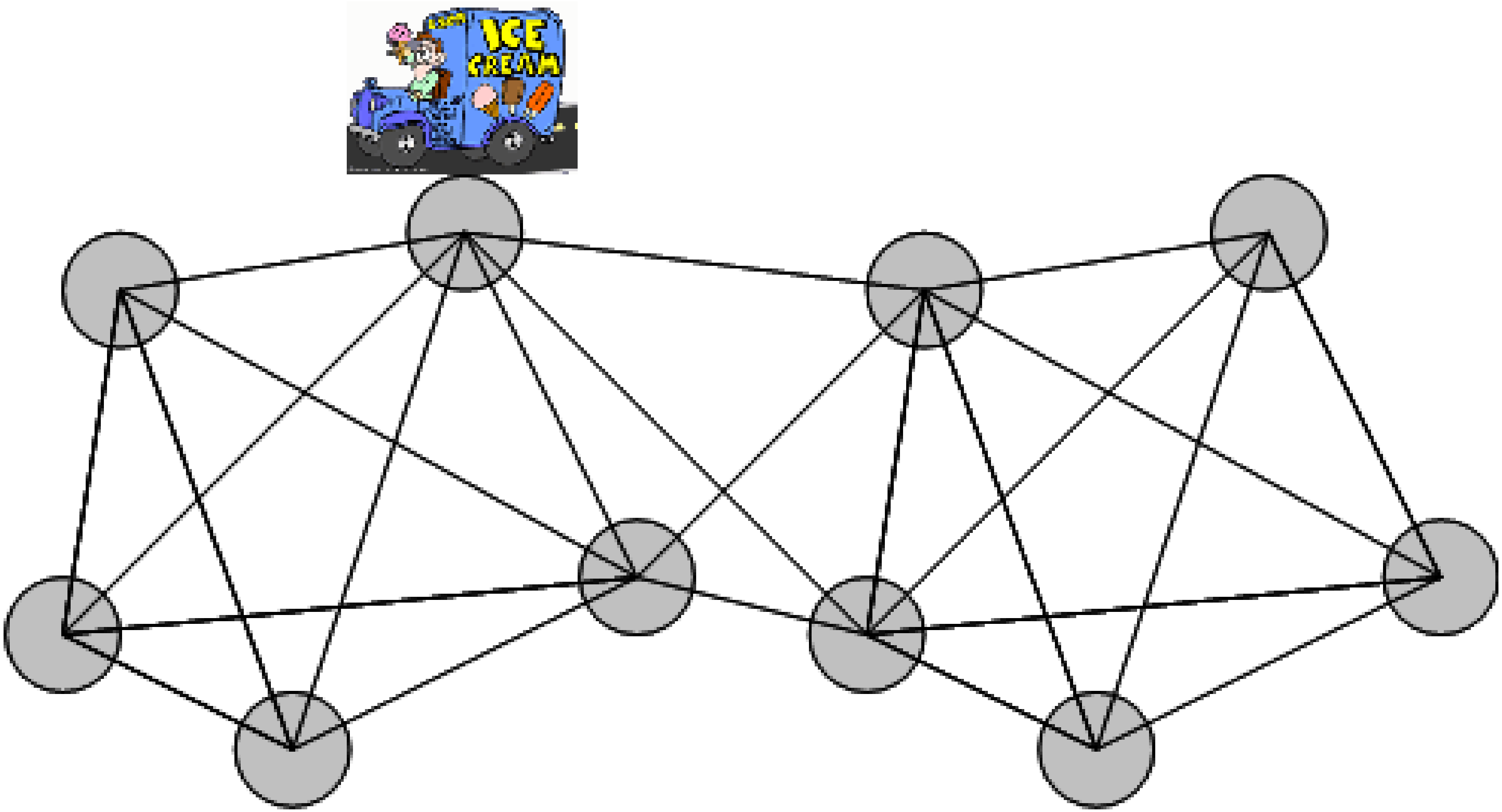
Parallel plans

- (A4 = 2, D1 = 3) => (A4 = 3) — independent
- (A2 = 3, B1 = 4) => (C1 = 5)

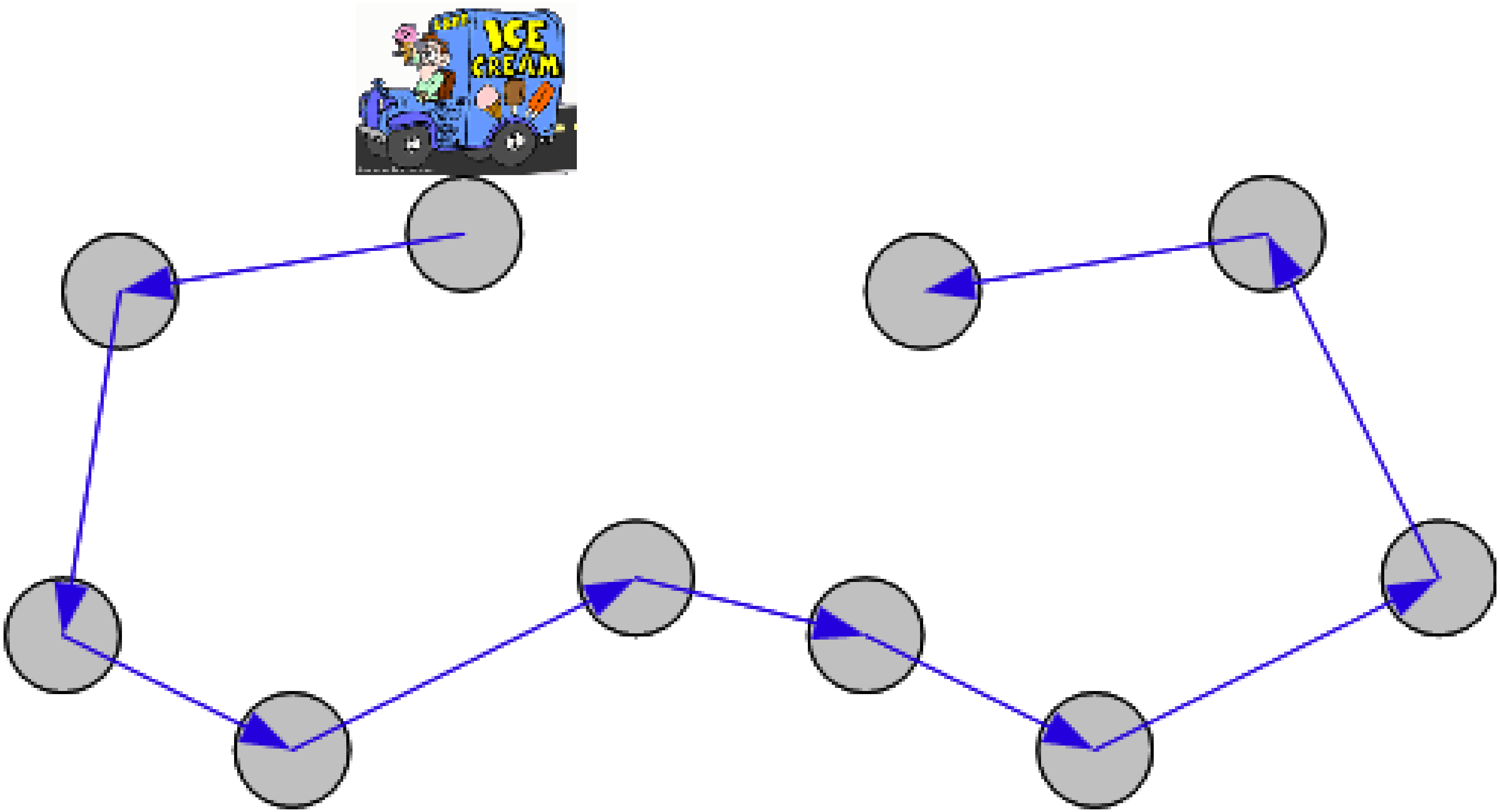
- (A4 = 2, D1 = 3) => (A4 = 3) — NOT independent
- (A2 = 3, B1 = 4) => (D1 = 5)

- (A4 = 2, D1 = 3) => (D1 = 1) — NOT independent
- (A4 = 2, B1 = 4) => (B1 = 5) — but parallelizable

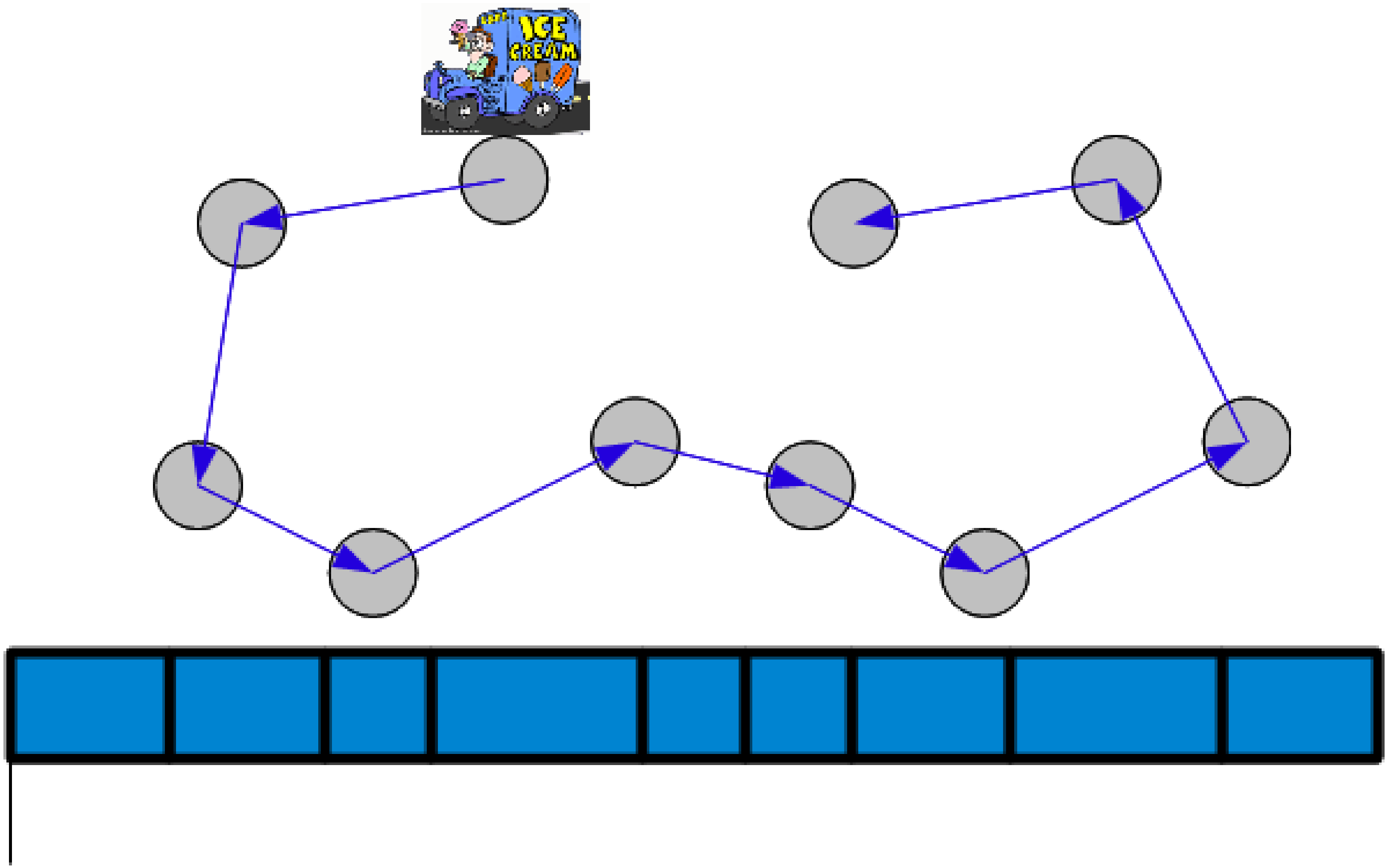
Parallel plans



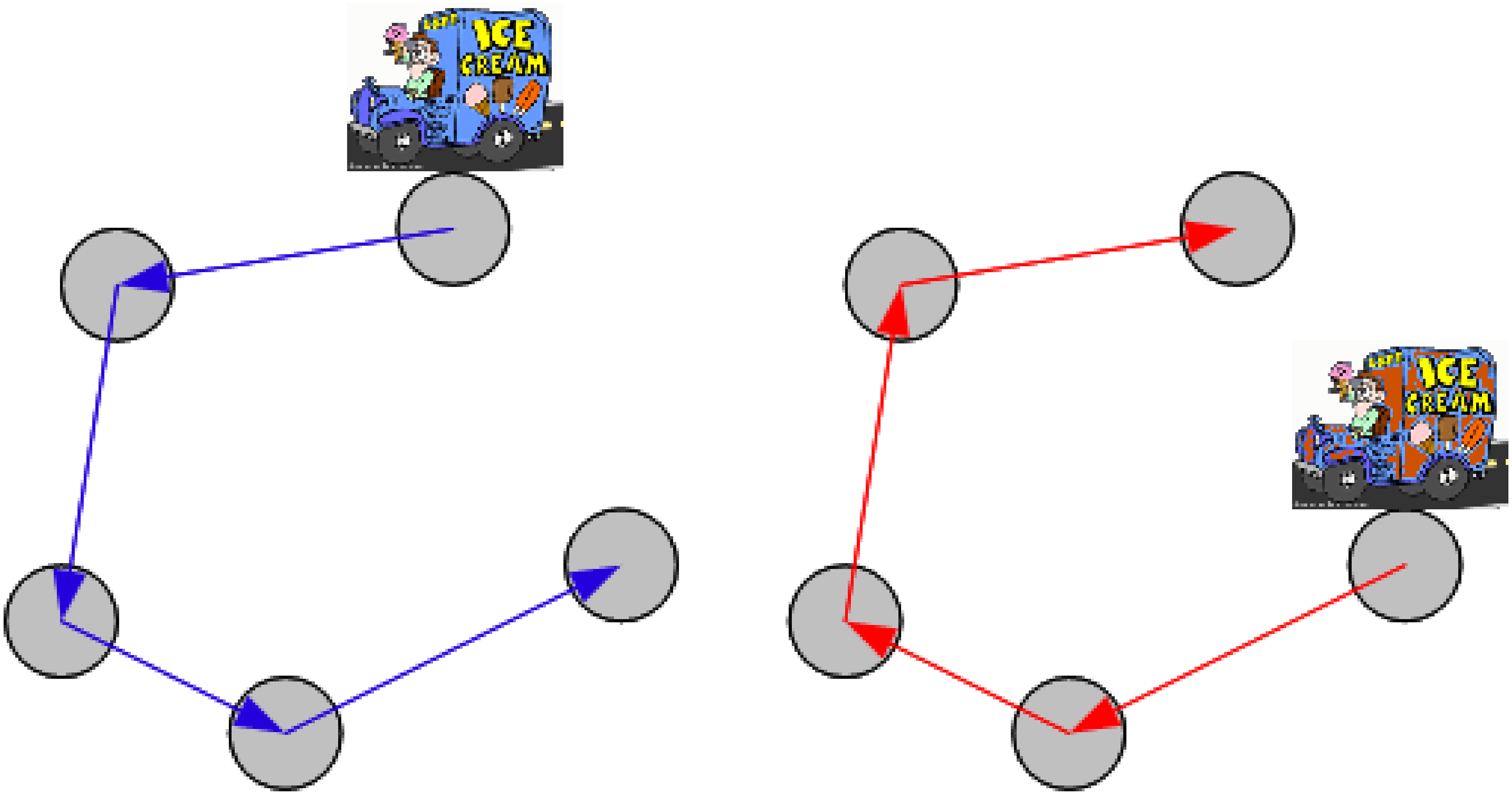
Parallel plans



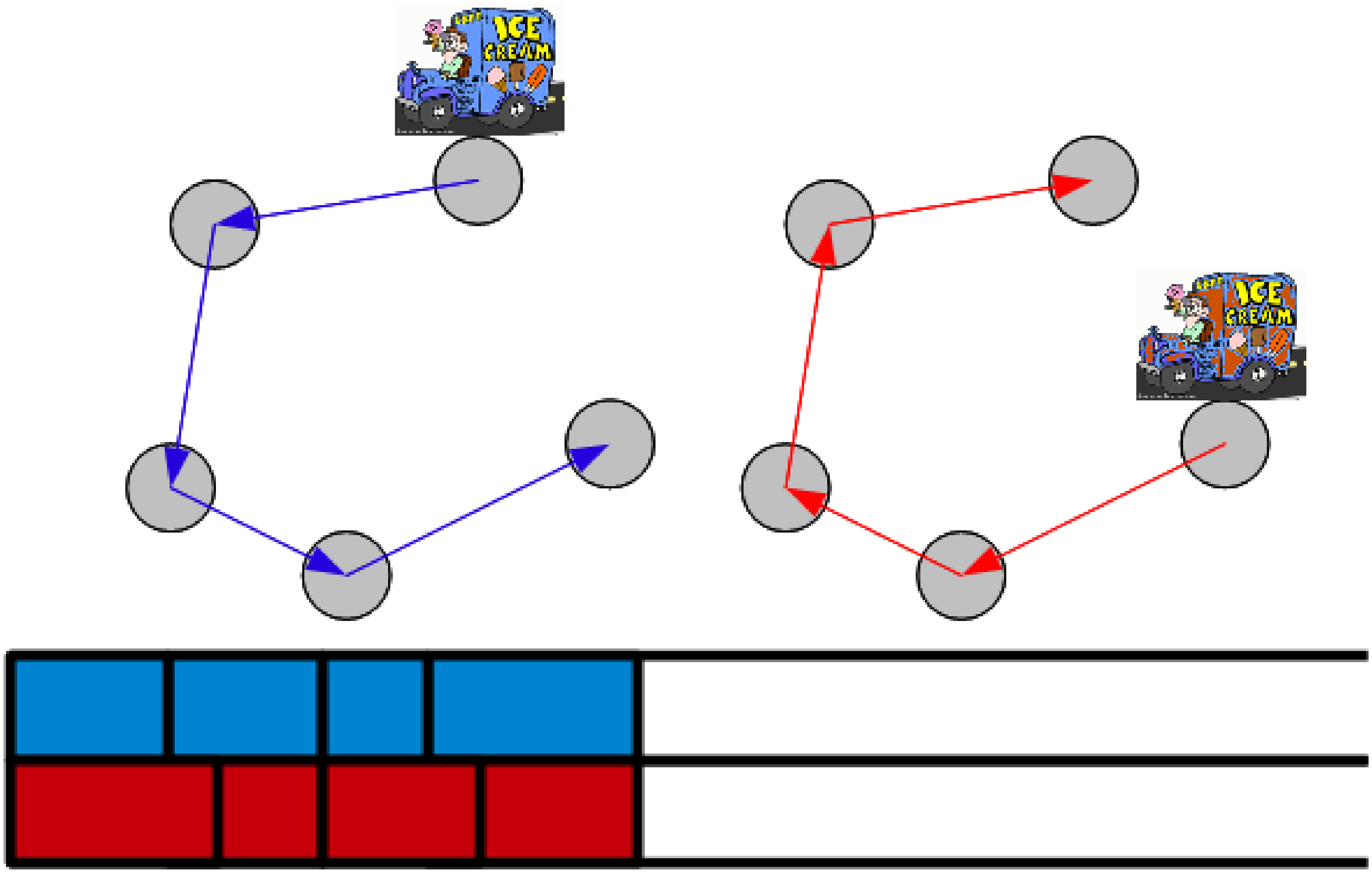
Parallel plans



Parallel plans



Parallel plans



Transitions

- Action can be seen as a set of transitions
- 3 kinds of transitions
  - Active
    - (A2 = 3, B2 = 4) => (B2 = 5, C2 = 6)
  - Prevailing
    - (A2 = 3, B2 = 4) => (B2 = 5, C2 = 6)
  - Mechanical
    - (A2 = 3, B2 = 4) => (B2 = 5, C2 = 6)

Reinforced encoding

- 3 kinds of SAT variables:
  - **Action variables**: a^t_i = *true* if
    action a_i occurs in the *t*-th parallel time step
  - **Assignment variables**: $b^t_{x=v}$ = *true* if
    variable *x* has value v at the end of *t*-th time step
  - **Transition variables**: $c^t_{x: d\text{->}e}$ = *true* if
    transition *x: d->e* occurs in the *t*-th time step
- + clauses ensuring correctness

Reinforced encoding - clauses

1. Only one value to each state variable
2. Used transitions imply values changes
3. Transitions' preconditions hold in the previous step
4. Value changes imply the use of proper transition
5. Using action imply using all its transitions
6. Transitions has to be supported by actions
7. Excluding compatible non-independent actions
8. Encoding the initial state and goal condition

- Usually shorter clauses than with other encodings
- Sophisticated reductions of the number of clauses

Other SAT encodings

- **Direct encoding**
  - Action based
  - Uses **action** and **assignment** variables
- **SASE encoding**
  - Transition based
  - Uses **action** and **transition** variables
- **$R^2\exists$-step encoding**
  - Uses different parallel semantics

Experimental results - coverage

Domain	Dir	SASE	Reinf	$R^2\exists$
barman	4	4	4	**8**
elevators	**20**	**20**	**20**	**20**
floortile	16	11	**18**	**18**
nomystery	**20**	10	**20**	6
openstacks	0	0	0	**15**
parcprinter	**20**	**20**	**20**	**20**
parking	0	0	0	0
pegsol	10	6	10	**19**
scanalyzer	14	12	**15**	9
sokoban	**2**	**2**	**2**	**2**
tidybot	**2**	**2**	**2**	**2**
transport	16	17	**18**	13
visitall	12	9	10	**20**
woodworking	**20**	**20**	**20**	**20**
Total	156	133	159	**172**

Conclusions & future work

- New encoding for planning as SAT
  - Outperforms other encodings on some domains
- Combination of *Direct* and *SASE* encoding
  - More variables may pay off
- Future work:
  - Decreasing the number of clauses
  - More compact way of encoding of the action interference constraints