HERIOT WATT UNIVERSITY

# P systems with active membranes operating under minimal parallelism

Pierluigi Frisco and Gordon Govan

# Summary: What it is all about

- P systems with active membranes
- Operating under minimal parallelism
- Using different sets of rules
- Solve **NP- & PP-complete** problems
- Simulate register machines

# Summary: What I'll show you today

Using a P system with active membranes operating under minimal parallelism to solve *k-SAT*

# Why: Why are we interested?

- P systems have been used to solve problems in different complexity classes
- and simulate different types of register machines
- Lots of features, rule sets and operating modes
- How does putting restrictions on how the P systems are used affect their efficiency and effectiveness?

# Why: Why are we interested?

**What are the necessary features required for each complexity class?**

# Features of P systems with active membranes

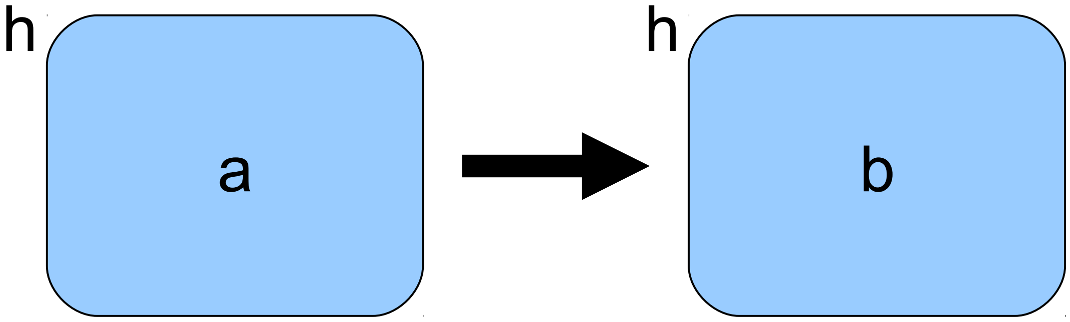
- Polarities
- Label rewriting
- Cooperative / catalytic evolution
- Compartment creation
- Elementary /  non-elementary membrane division

# Features: Rule *(a)*

Rewrites the multiset of a compartment

*b* is a string of symbols

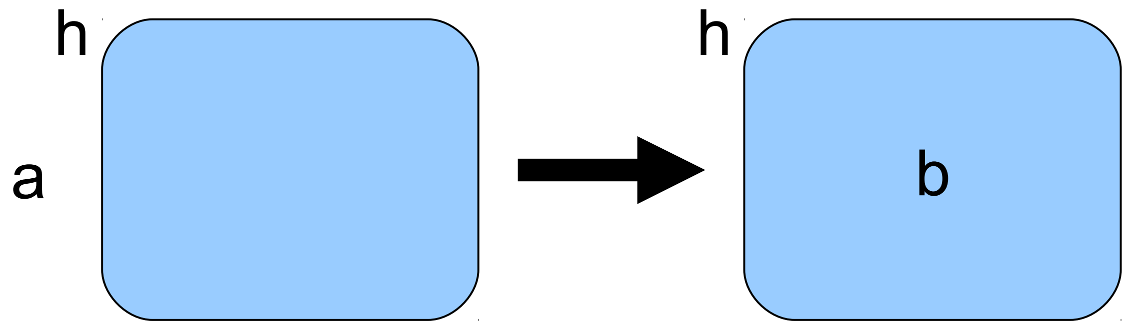
$$[a \rightarrow b]_h$$



# Features: Rule *(b)*

Move an object into a compartment

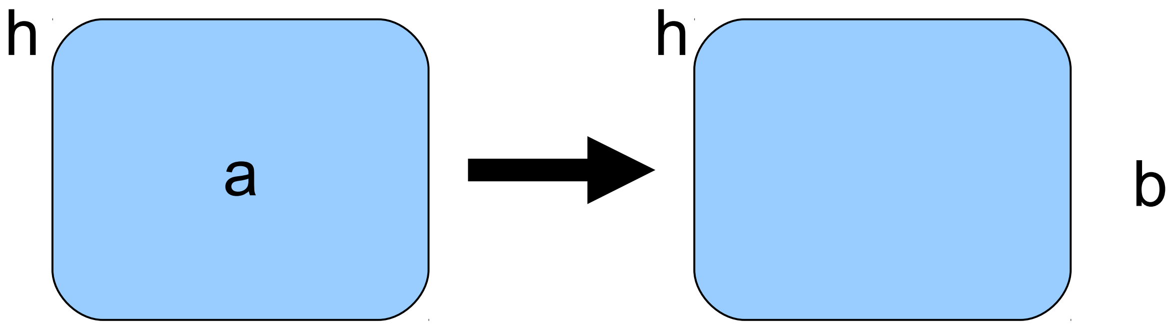
$$a\,[\;]_h \rightarrow [\,b\,]_h$$



# Features: Rule *(c)*

Move an object out of a compartment

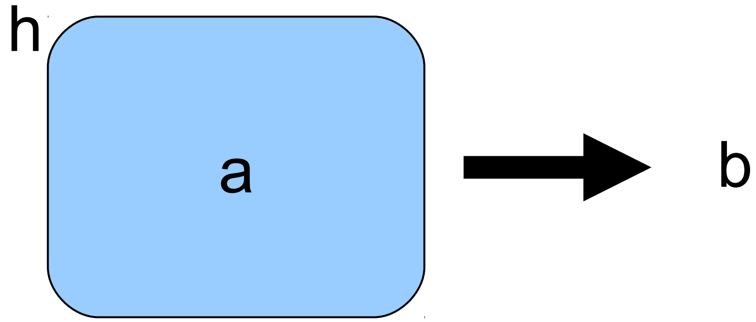
$$[\, a \,]_h \rightarrow [\;]_h\, b$$



# Features: Rule *(d)*

Remove a compartment

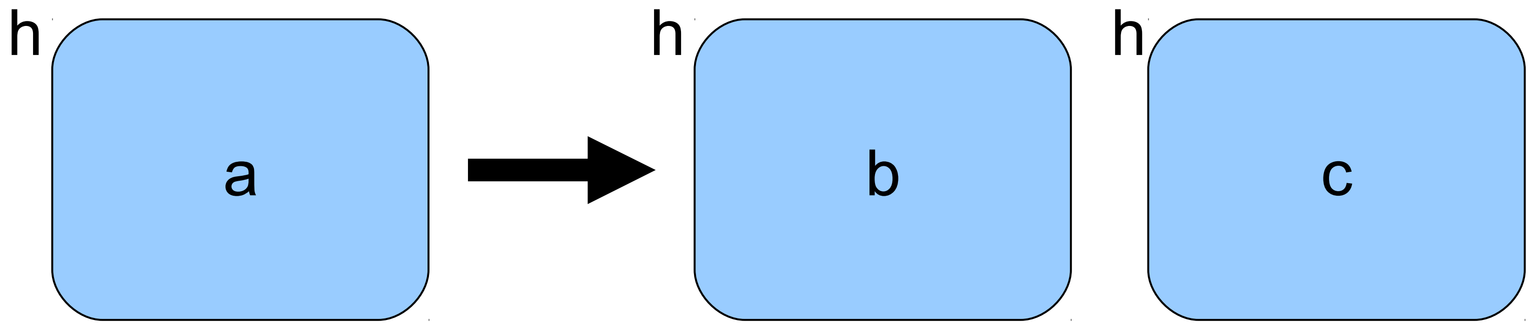
$$[\,a\,]_h \rightarrow b$$



# Features: Rule *(e)*

Divide a compartment

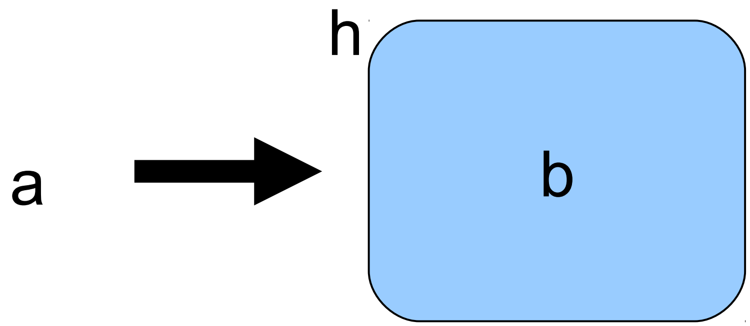
$$[\, a \,]_h \rightarrow [\, b \,]_h [\, c \,]_h$$



# Features: Rule *(g)*

Create a compartment

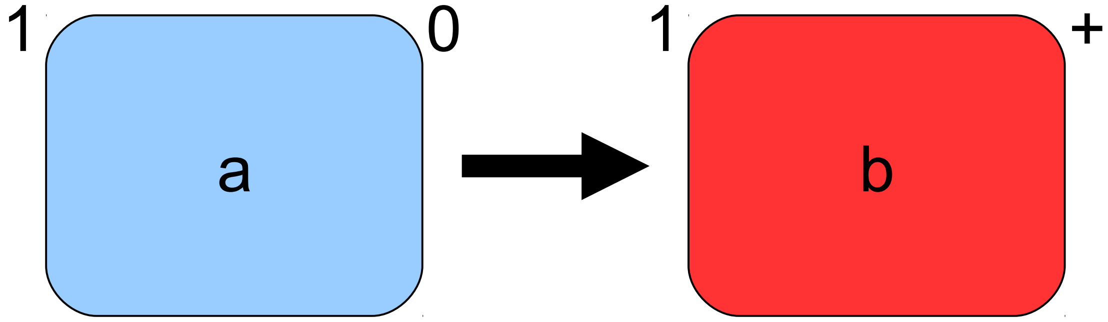
$$a \rightarrow [\, b \,]_h$$



# Features: Polarities

An example of a rule that changes polarity

$$[\,a\,]_1^0 \rightarrow [\,b\,]_1^+$$



# Features: **Minimal Parallelism**

In each transition for each compartment at least one rule is applied at least once where possible

# Prior: What has been done before

| Class | Operating mode | Polarities | Label rewriting | Membrane division | Evolution rules | Rules used |
|---|---|---|---|---|---|---|
| NP | Minimal | Yes | No | Non-elementary | | *(a)-(e)* |
| NP | Minimal | No | Yes | Non-elementary | | *(a) (c) (e)* |
| NP | Minimal | No | No | Non-elementary | Cooperative | *(a)-(c) (e)* |
| NP | Minimal | No | No | Elementary | Cooperative | *(a) (c) (e)* |

# What did we do?

- P system with active membranes acting under minimal parallelism without polarities

- Solve *k-SAT - a* **NP**-complete problem

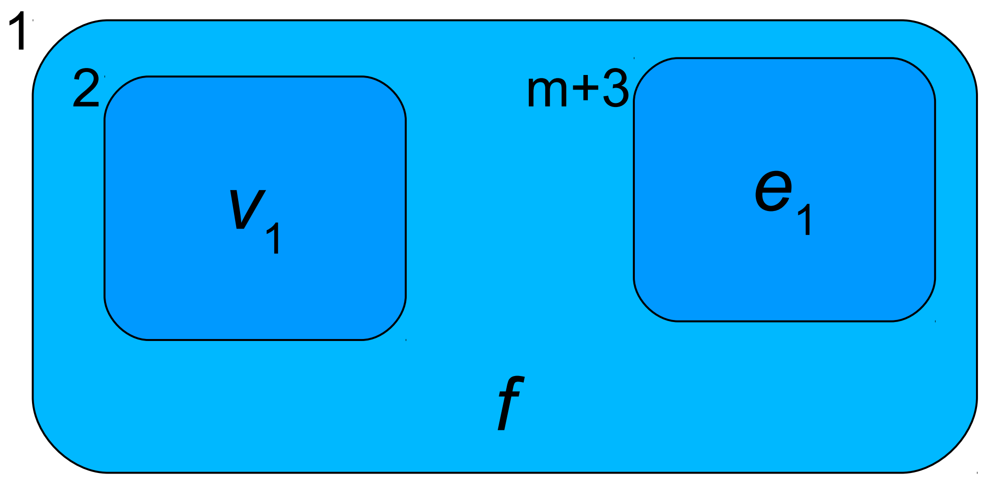
- Using rules of type *(a)*, *(b)*, *(c)*, *(e)*, and *(g)*

# What is *k-SAT*?

Given a boolean formula $\psi$ with $m$ clauses in conjunctive normal form where each clause is a disjunction of literals $v_i$ or $\overline{v}_i$, $1 \leq i \leq n$

HERIOT WATT UNIVERSITY

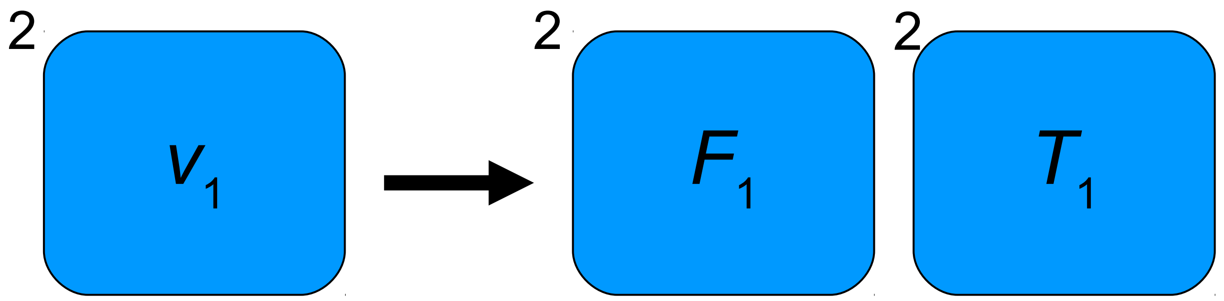
# How: *Construct the system*



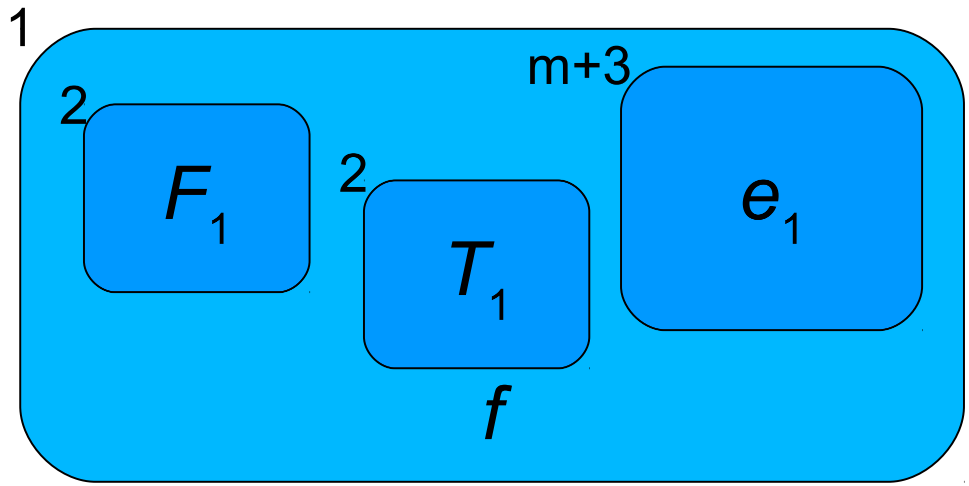
HERIOT WATT UNIVERSITY

# How: Rule 1

$$[\,v_i\,]_2 \rightarrow [\,F_i\,]_2 [\,T_i\,]_2$$



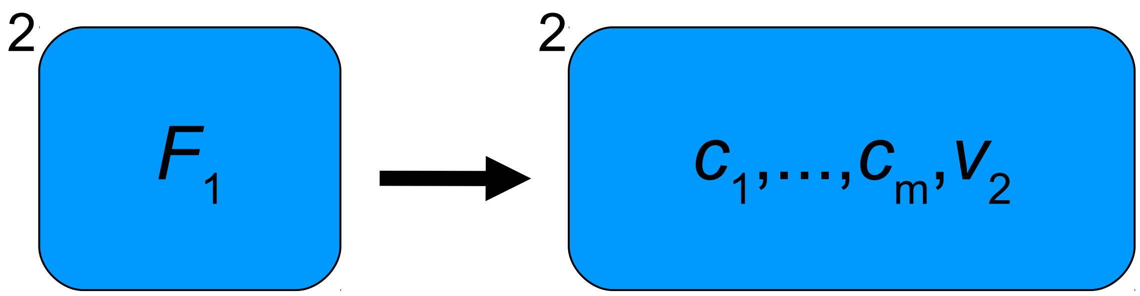
# How: After rule 1



# How: Rules 2 and 3

- Rules 2 and 3 use two functions: *true* and *false*
- Both functions are from $\{v_1,\ldots,v_n\}$ to $\mathbf{P}\{c_1,\ldots,c_m\}$
- $true(v_i)$ returns the set of clauses verified by $v_i$
- $false(v_i)$ returns the set of clauses verified by $\overline{v}_i$

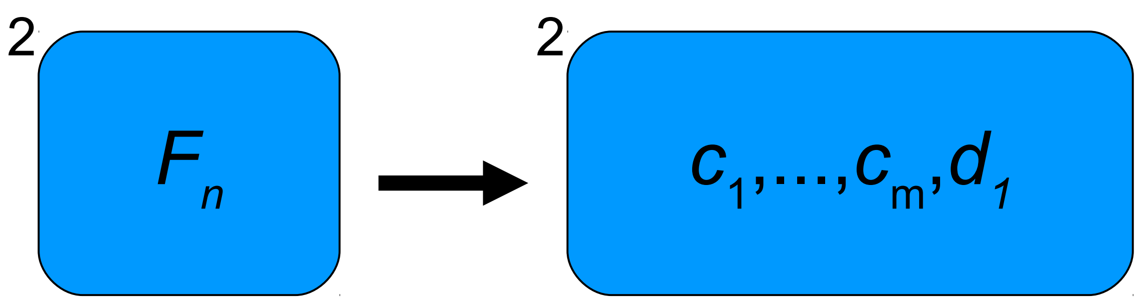
$$[F_i]_2 \rightarrow [false(v_i)\, v_{i+1}]_2 \qquad [T_i]_2 \rightarrow [true(v_i)\, v_{i+1}]_2$$



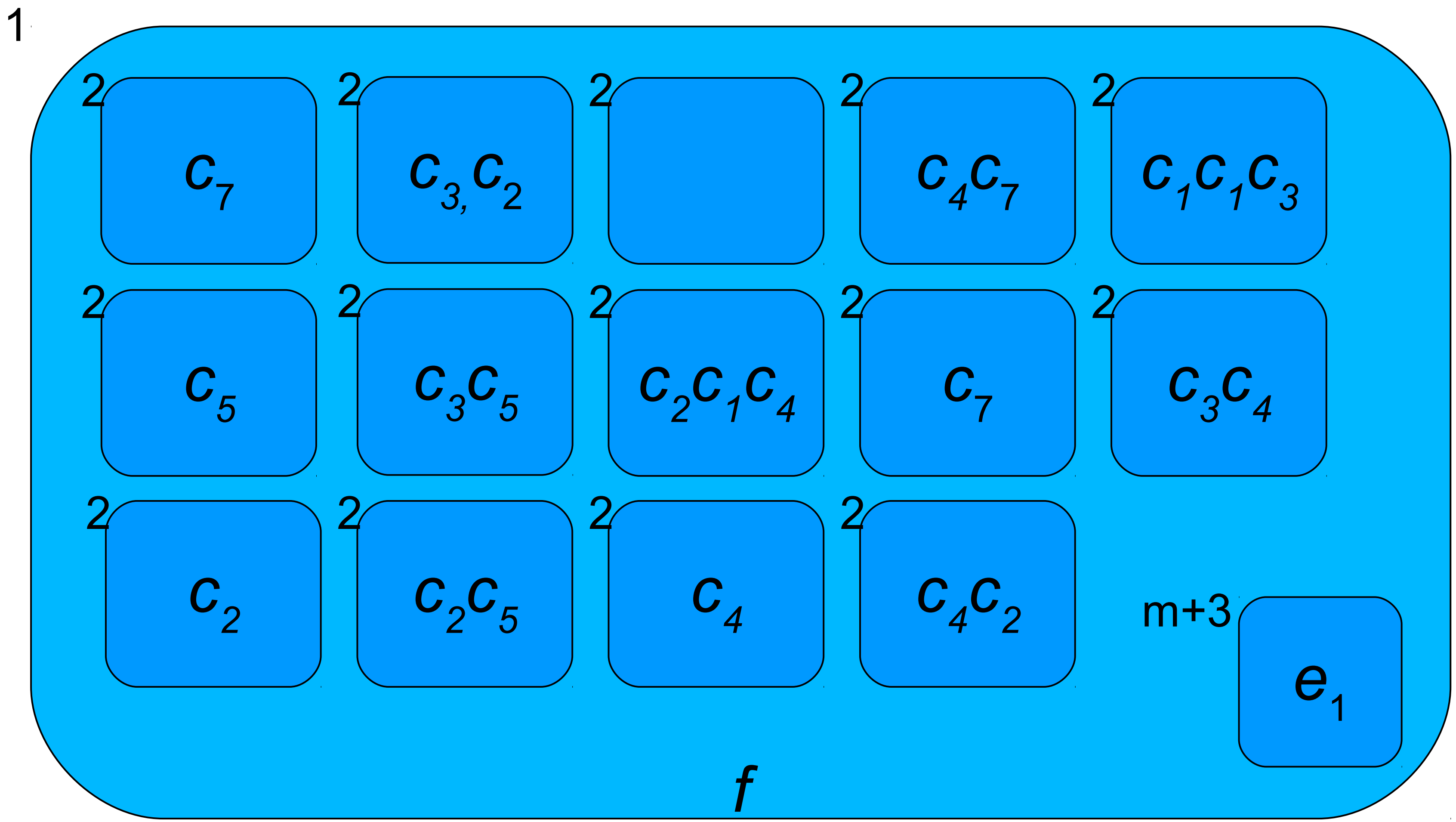
# How: Rules 4 and 5

- Rules 4 and 5 similar to 2 and 3
- But only for $F_n$ and $T_n$
- Also leaves a $d_1$

$$[F_n]_2 \rightarrow [false(v_n)\, d_1]_2 \qquad [T_n]_2 \rightarrow [true(v_n)\, d_1]_2$$

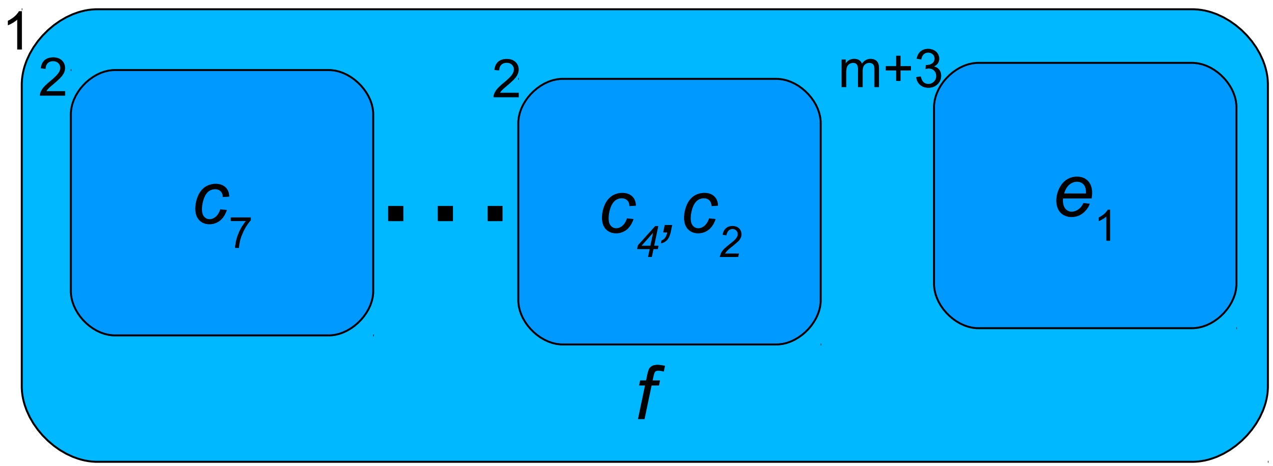


# How: After rules 1 through 5



# How: After rules 1 through 5

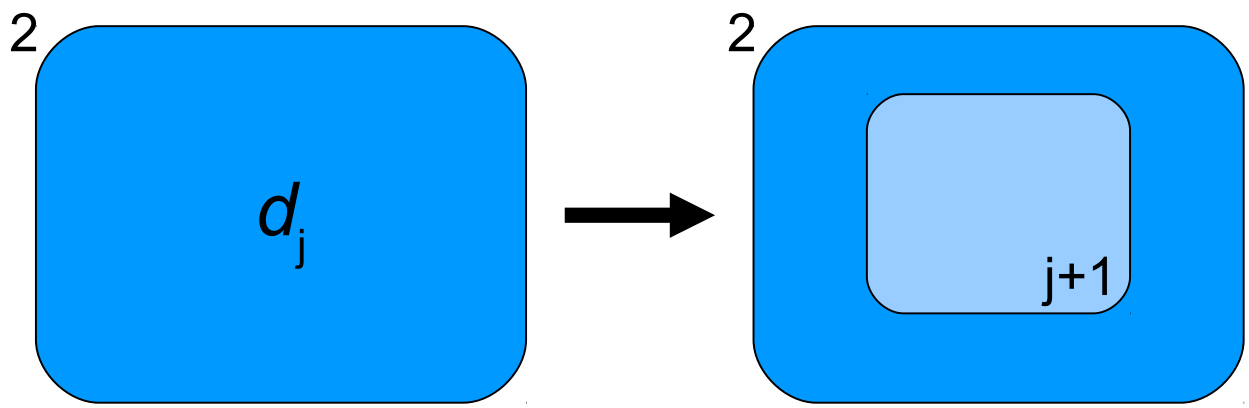
- There are $2^n$ copies of compartment 2
- Each with a subset of $\{c_1,\ldots,c_m\}$
- The clauses which that assignment of variables satisfies



# How: Rule 6

- Rule 6 uses $d_j$ to create a compartment i+2
- Used for checking if $c_j$ exists

$$[d_j]_2 \rightarrow [[\,]_{j+2}]_2$$

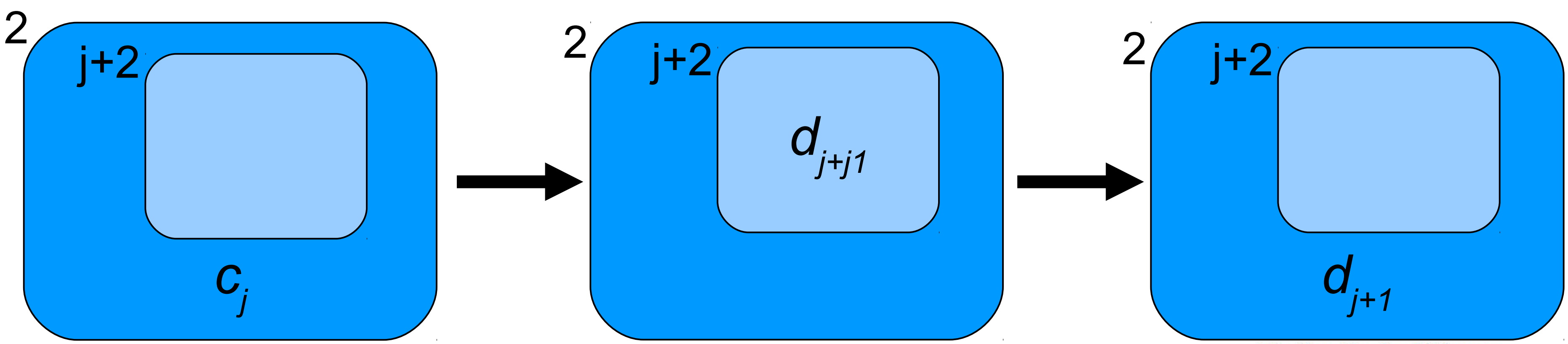


# How: Rules 7 and 8

- Rules 7 and 8 check to see if $c_i$ exists
- If $c_j$ exists then $d_{j+1}$ will be created

$$c_j [\,]_{j+2} \rightarrow [d_{j+1}]_{j+2}$$

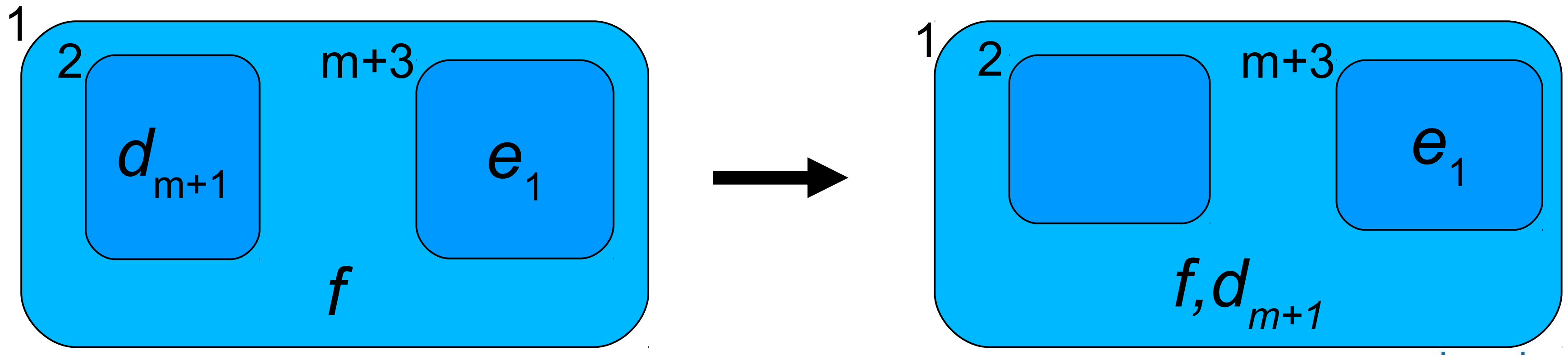
$$[d_{j+1}]_{j+2} \rightarrow [\,]_{j+2} d_{j+1}$$



# How: Rule 9

- If a compartment satisfies $\psi$ then there will be a $d_{m+1}$
- Rule 9 makes any $d_{m+1}$ pass from the 2 compartments into the 1 compartment

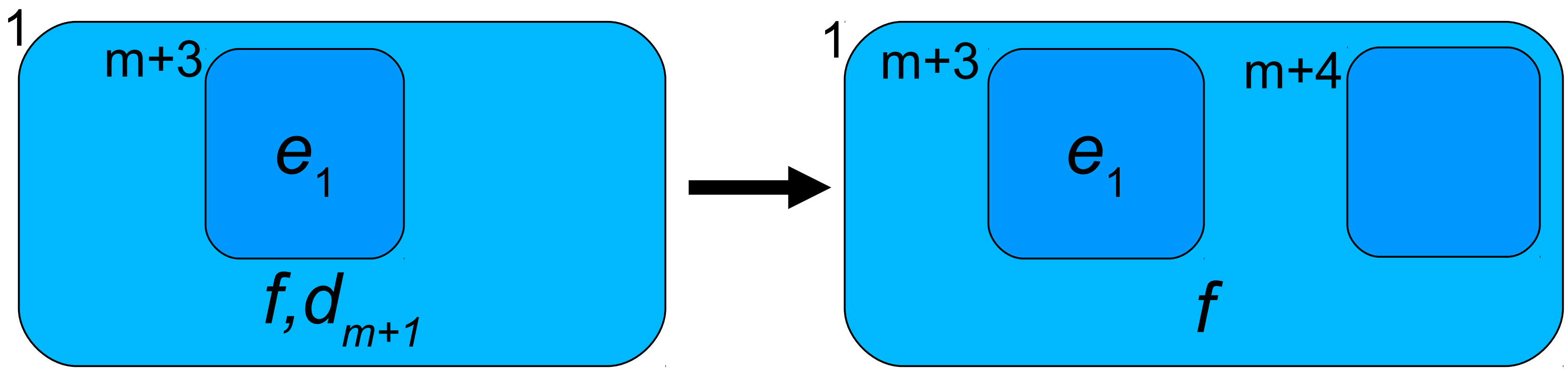
$$[d_{m+1}]_2 \rightarrow [\ ]_2 d_{m+1}$$



# How: Finishing it off

- If there is a $d_{m+1}$ in compartment 1 then -
- Rule 12 will create a *m+4* compartment

$$\left[d_{m+1}\right]_1 \rightarrow \left[\left[\;\right]_{m+4}\right]_1$$

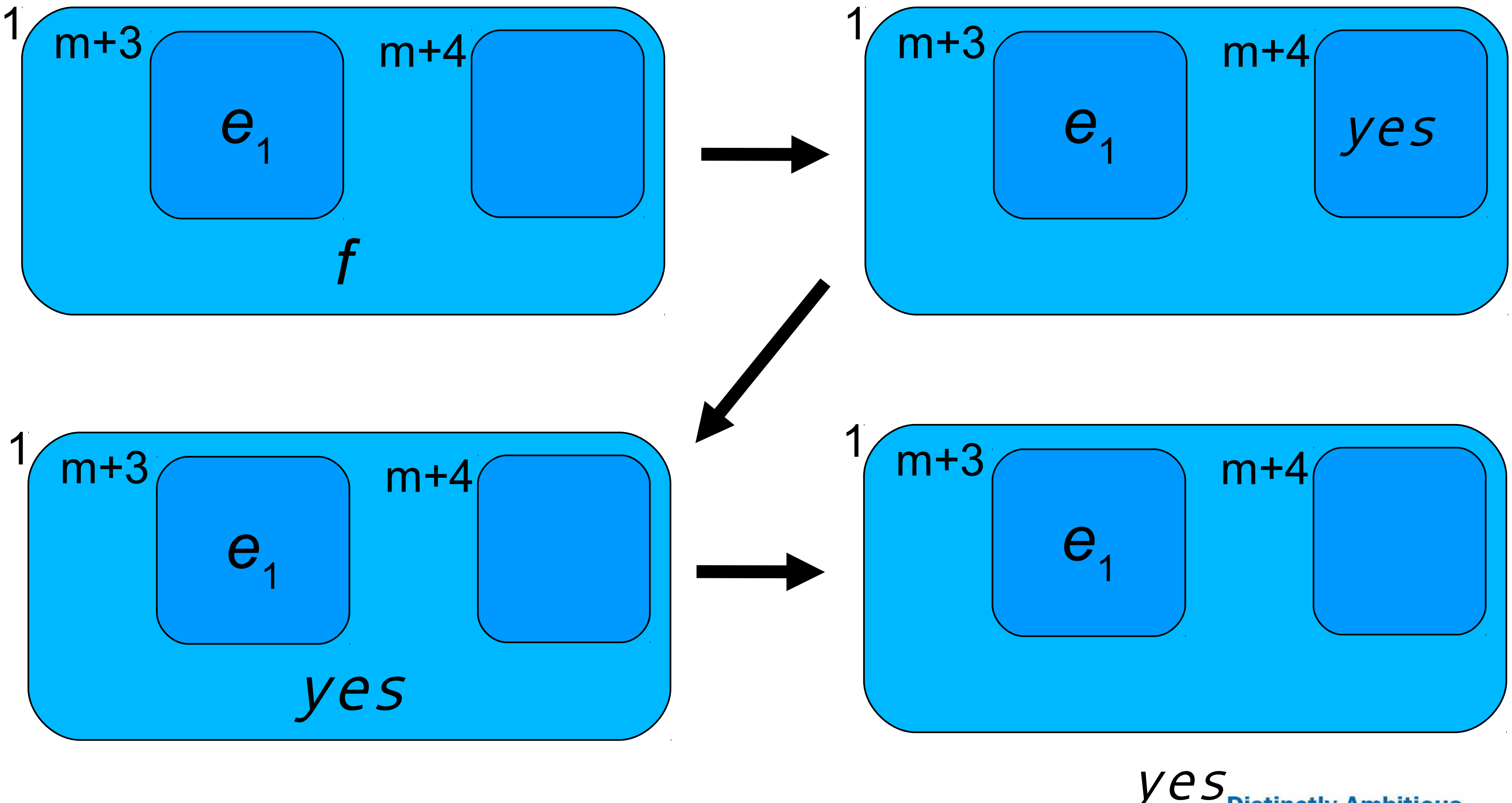


# How: Finishing it off

Rule 13, 14, and 10 will:

- Cause *f* to enter m+4 and become *yes*
- *yes* will pass back into 1
- *yes* will then pass into the environment
- The system will then halt

# How: Finishing it off

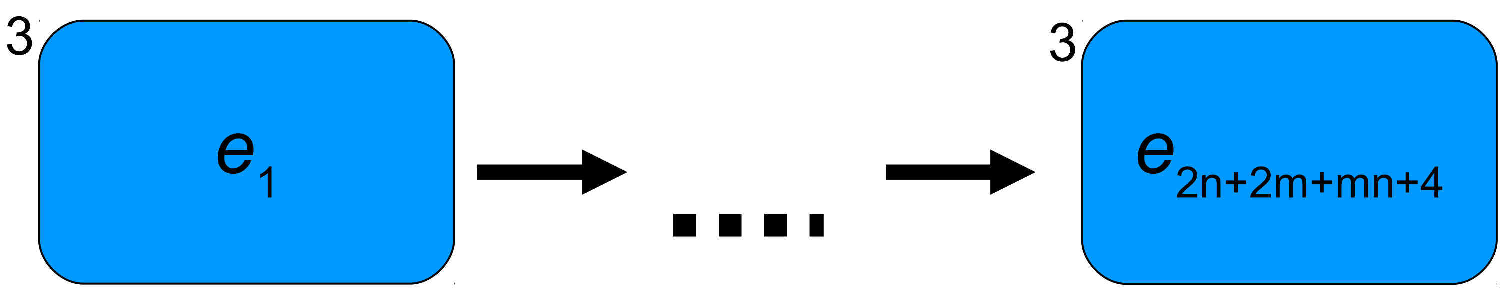


# How: What is this compartment m+3 up-to?

- Compartment 3 is busy at work
- Acting as a clock
- Rule 11 increases the subscript of *e*

$$[e_i]_3 \rightarrow [e_{i+1}]_3$$

- for $1 \leq i \leq 2n+2m+mn+3$

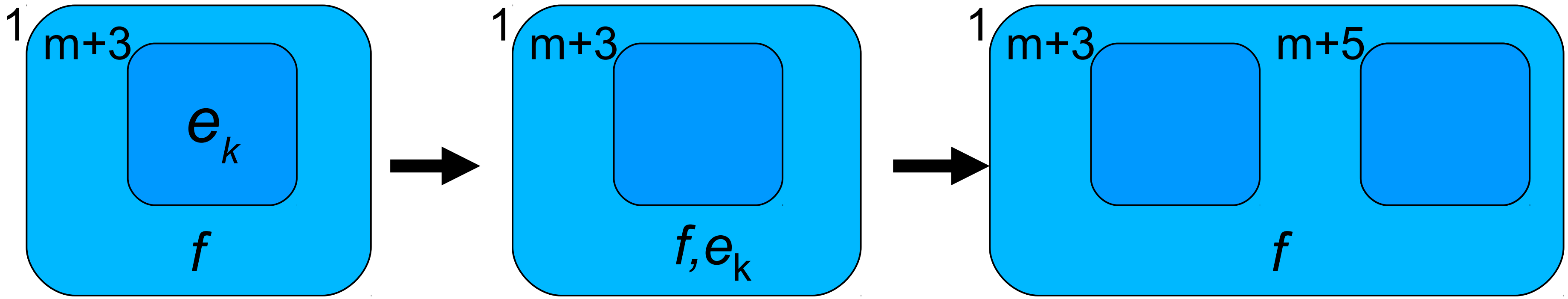


# How: Rules 15 and 16

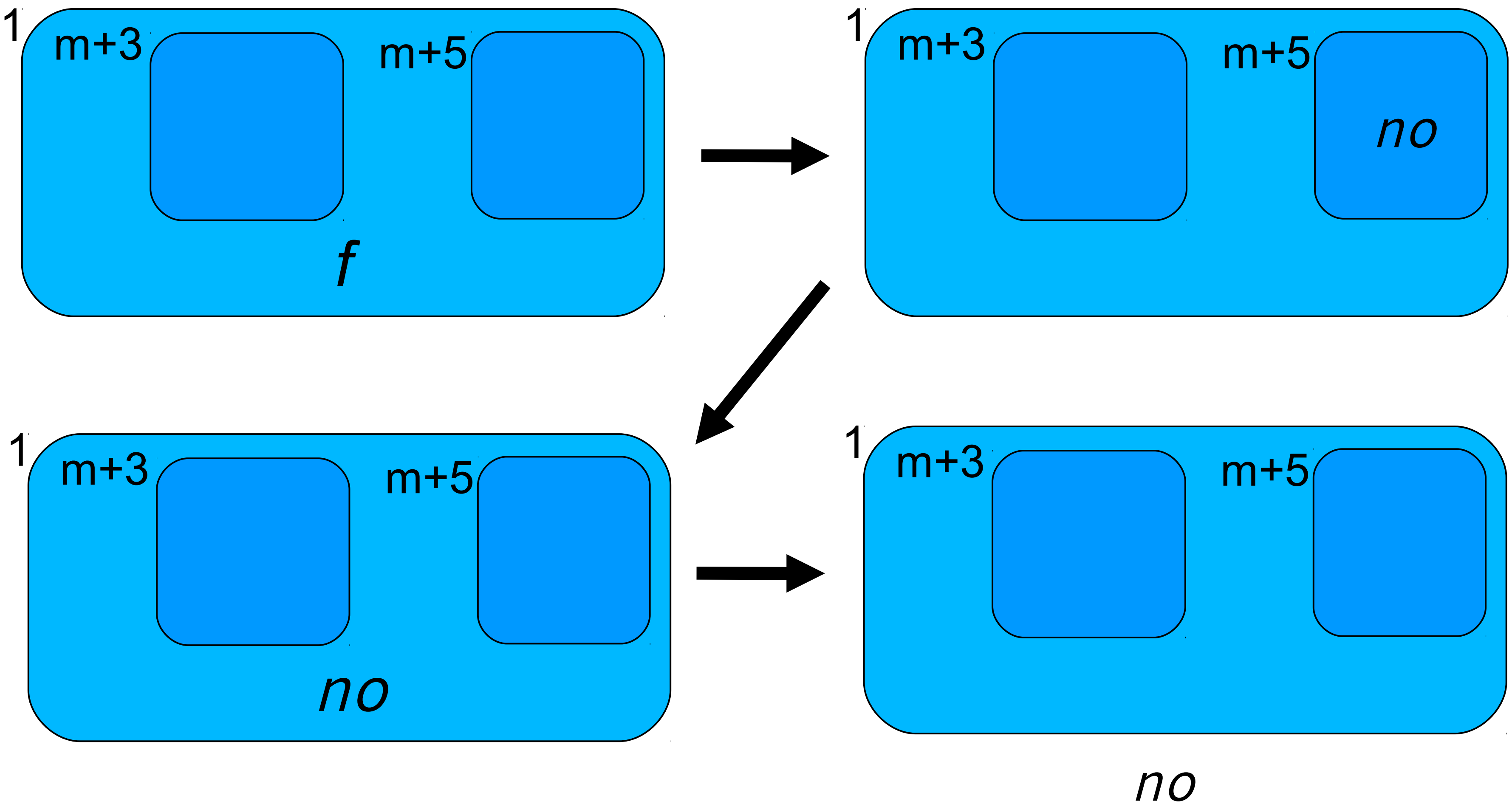
When it has finished counting, $k=2n+2m+mn+4$:

- The *e* will pass into compartment 1
- It will then create a *m+5* compartment

$$[e_k]_{m+3} \rightarrow [\,]_{m+3} e_k$$
$$[e_k]_1 \rightarrow [[\,]_{m+5}]_1$$



# How: Rules 17 and 18



# Results: After halting

- This system will take up to $2n+2m+mn+9$ to run
- Bound on $mn$
- *yes* will pass into the skin compartment if $\psi$ is satisfiable
- *no* will pass into the skin compartment otherwise
- Makes no assumptions on $k$

# Results: Other operational modes

- The system can be run under different operational modes
- The system will still work under both maximal strategy and maximal parallelism
- The system will perform differently for maximal parallelism and maximal strategy
- Would not work asynchronously

# Results: New table

| Class | Operating mode | Polarities | Label rewriting | Membrane division | Evolution rules | Rules used |
|---|---|---|---|---|---|---|
| NP | Minimal | Yes | No | Non-elementary | | *(a)-(e)* |
| NP | Minimal | No | Yes | Non-elementary | | *(a) (c) (e)* |
| NP | Minimal | No | No | Non-elementary | Cooperative | *(a)-(c) (e)* |
| NP | Minimal | No | No | Elementary | Cooperative | *(a) (c) (e)* |
| **NP** | **Minimal** | **No** | **No** | **Elementary** | | **(a)-(c) (e) (g)** |

# Results: PP-complete problems

- Previously solved by P systems with active membranes operating under maximal parallelism

- We use minimal parallelism to solve MAJORITY-*SAT: a* **PP**-complete problem

- Use a similar approach as for k-Sat

- Using polarities and rules of type *(a)*, *(b)*, *(c)*, and *(e)*

- System runs in linear time in regards to *mn*

# Results: Register Machine

- Previously simulated by P systems with active membrane operating under minimal parallelism

- But using polarities, label rewriting rules, or cooperative evolution rules

- We use none of these features

- Use membrane creation and dissolution instead

- Rules of type *(a)*, *(b)*, *(c)*, *(d)*, and *(g)*

# Conclusions:

- Solved problems using minimal parallelism

- Used different rules and features from previous work

- We found a set of rules that are able to solve these problems

- But could these sets be smaller?

# End: Thanks for listening

**Any questions?**