Smart Graphics: Methoden 5 graph drawing, operator based planning

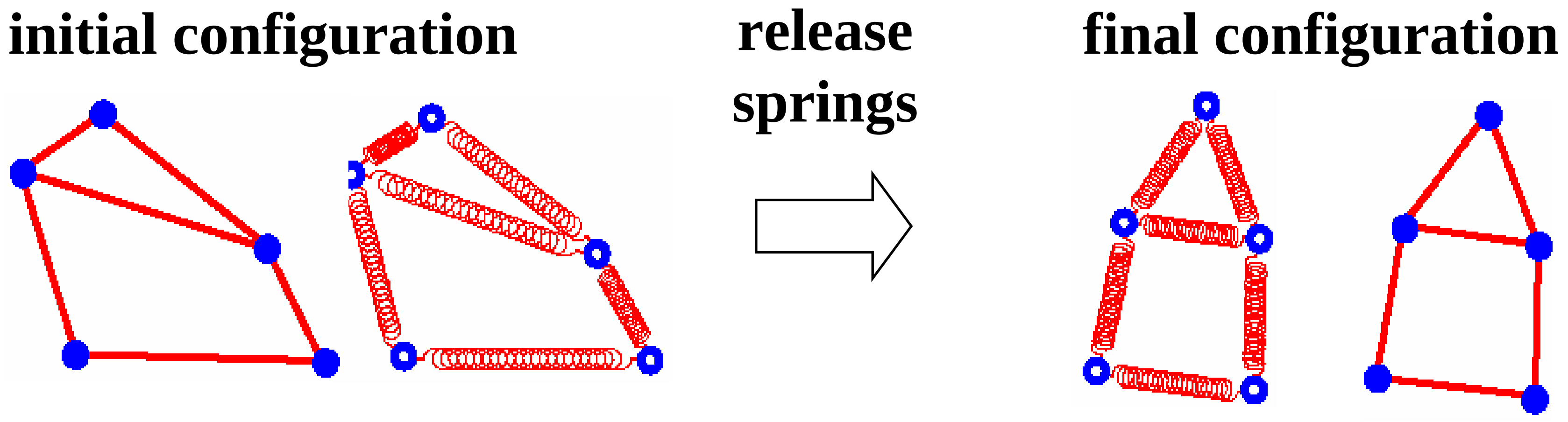
Vorlesung „Smart Graphics”

Andreas Butz, Otmar Hilliges

Dienstag, 13. Dezember 2005

Force-directed methods

- Hirsch formulated a successful solution technique for the PLFP problem using a force analogue, where the repulsion of a label position was estimated from the overlap regions
- graph drawing can also be done by force-directed methods



Framework for force-directed drawing

Force-directed drawing approaches comprise:

- **MODEL**

  A system of attractive and repulsive forces which is mapped onto the edges and vertices of a graph according to the required aesthetic criteria and constraints

- **SOLUTION TECHNIQUE**

  The technique for finding positions for every vertex of the graph such that the system of forces in an equilibrium state (i.e. the sum of the forces at all vertices is zero)

A simple force-directed model

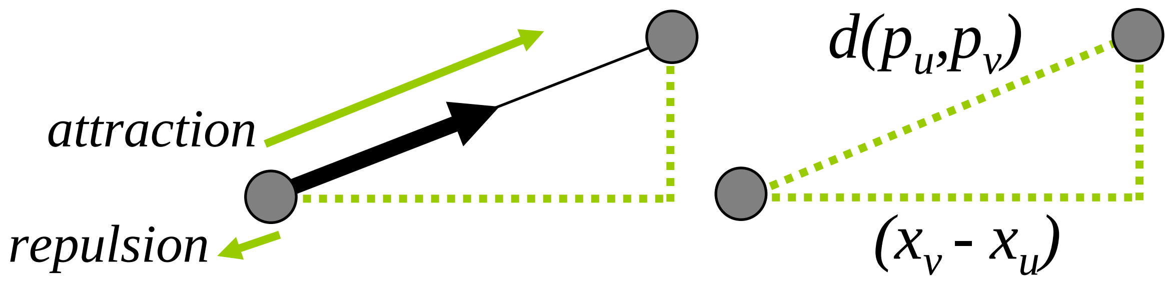
- a simple force-directed model using spring and electrical forces:
  - edges are modelled as springs and vertices as equally charged electrical particles

$$F(v) = \sum_{(u,v)\in E} f_{uv} + \sum_{(u,v)\in V\times V} g_{uv}$$

- force on a vertex, F_{uv} is given by:
  - f_{uv} is proportional to the length of the spring
  - g_{uv} is the electrical repulsion between two particles
- thus, the x-component of the force is given by:

$$F_x(v) = \sum_{(u,v)\in E} k^1_{uv}(d(p_u, p_v) - l_{uv}) \frac{x_v - x_u}{d(p_u, p_v)} + \sum_{(u,v)\in V\times V} \frac{k^2_{uv}}{(d(p_u, p_v))^2} \frac{x_v - x_u}{d(p_u, p_v)}$$

$$F_x(v) = \sum_{(u,v)\in E} k^1_{uv}(d(p_u, p_v) - l_{uv})\frac{x_v - x_u}{d(p_u, p_v)} + \sum_{(u,v)\in V\times V} \frac{k^2_{uv}}{(d(p_u, p_v))^2}\frac{x_v - x_u}{d(p_u, p_v)}$$



- $p_u=(x_u, y_u)$ and $p_v=(x_v, y_v)$ are the positions of vertices u and v
- $d(p_u,p_v)$ is the distance between *u* and *v*
- k^1 is the spring constant, the stiffness of the edge springs
- k^2 is the electrical repulsion constant
- l_{uv} is the unstretched spring length, when $d(p_u,p_v) = l_{uv} \rightarrow$ no force
- $(x_v - x_u) / d(p_u,p_v)$ is the x-component of the repulsion and spring force

Solution techniques and other models

- performing a local search is the simplest solution technique:
  1. randomly place the vertices
  2. loop until all forces are below a threshold:
     - compute the forces on each vertices
     - move each vertex in the direction of, and a distance proportional to, its force
- other force models:
  - barycenter method:
    - no repulsion, l_{uv} to zero for all edges
    - “nail down” the positions of at least three vertices
  - graph-theoretic distances:
    - tries to make the Euclidean distance between all pairs of vertices *u* and *v* (not just directly connected vertices) $d(p_u,p_v)$ proportional to the graph theoretic distance $\delta(u,v)$ (the shortest path length from *u* to *v*)

General energy functions

- simple model: energy function is a continuous function of the locations of the vertices
- many important aesthetic criteria are not continuous:
  - number of crossings
  - number of horizontal and vertical edges
- to optimize several criteria simultaneously they can be linearly combined:

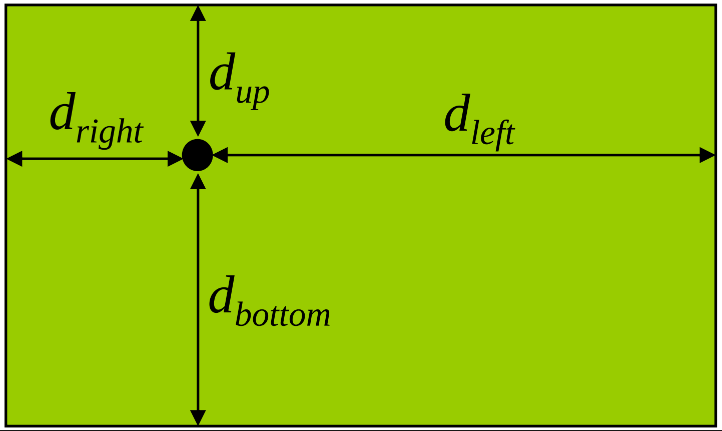
  $\eta = \lambda_1\eta_1 + \lambda_2\eta_2 + \lambda_3\eta_3 + \lambda_4\eta_4 + \dots$
- for example, (Davidson and Harel, 1996) use the following energy function:

$$\eta_1 = \sum_{u,v \in V} \left(1/d(p_u,p_v)^2\right) \dots \text{as before}$$

$$\eta_2 = \sum_{(u,v) \in E} (d(p_u, p_v))^2 \quad \dots \text{as before}$$

$$\eta_3 = the\, number of\; edge crossings$$

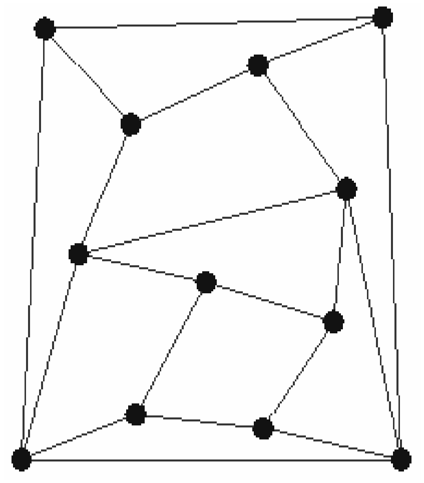
$$\eta_4 = \sum_{u \in V} \left(\frac{1}{d_{left}^2} + \frac{1}{d_{right}^2} + \frac{1}{d_{top}^2} + \frac{1}{d_{bottom}^2} \right)$$

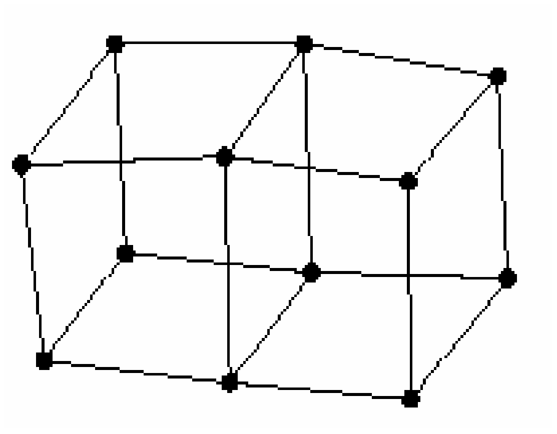


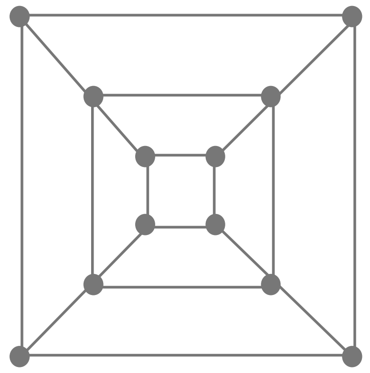
Simulated annealing

- general energy functions require more global optimization techniques, for example, (Davidson and Harel, 1996) used simulated annealing:
  1. choose an initial configuration
  2. repeat some fixed number of times:
     - choose a new configuration C* from the neighbourhood of C
     - if ($\eta^* < \eta$) set C to C*, else set C to C* with a probablity $e^{(E-E^*)/T}$
  3. decrease the temperature T
  4. if the termination rule is satisfied then stop, else continue from step 2
- configuration choice was by random placement of a single vertex within a (decreasing) radius from its current position, thus there is a cooling schedule applied to the perturbation of vertices
- Contrast this with the Christensen's application of SA to PFLP…

(Davidson and Harel, 1996)


Crossings heavily penalised


Crossings allowed


Symmetric crossing-free drawing not produced

Evaluation of force-based methods

Strengths

- relatively simple to implement
- smooth evolution of the drawing into the final configuration helps preserving the user’s mental map
- can be extended to 3D
- often able to detect and display symmetries
- works well in practice for small graphs with regular structure
- Smooth transitions come for free

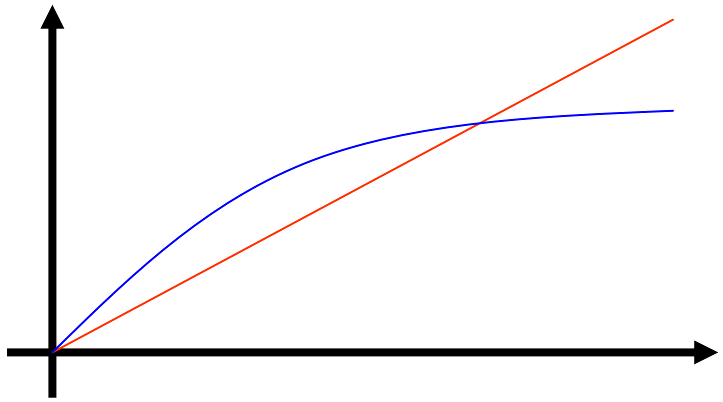
Weaknesses

- slow running time (depending on solution technique, but in general only up to hundreds of vertices)
- few theoretical results on the quality of the drawings produced
- difficult to extend to orthogonal and polyline drawings
- limited ability to add constraints and maintain constraints

Anytime-algorithms

- Provide a result after arbitrary running time
- Result quality increasing monotonously
  - Measurement function for result quality
  - Constant improvement of current results
  - Mostly incremental approaches
  - interruptability
- Can be applied to different base algorithms

- Quality / Time tradeoff



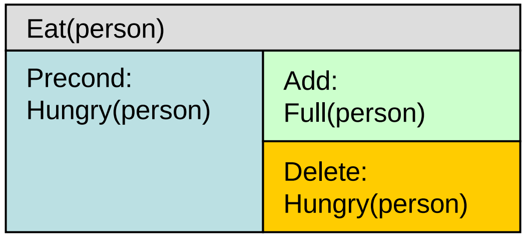
Operator-based planning

STRIPS Stanford Research Institute Problem Solver (1972)

- Originally used for robot action planning
- Given a state and an action:
  - Can the action be executed?
  - What propositions are (in)valid after the action?
- Closed world assumption:
  - We know all true statements
  - All unknown statements are false
- Description of initial state and goal state
  - Sets of positive facts
- Set of STRIPS-operators with 3 parts:
  - Preconditions: what needs to hold, such that the op. can be ex.?
  - Add list: what holds after the op. is ex.?
  - Delete list: what doesn’t hold after the op. is ex.?
- Planning problem: how to get from initial to goal state

Example STRIPS operator

- Operator name: Eat
  - Parameter: a person



- Can only be applied if person is hungry
- Removes the fact that person is hungry
- Adds the fact that person is full

- Remember: only positive facts!

Christmas evening

- Initial state
  - Hungry(children)
  - Want(children,gifts)

- Goal state
  - Happy(children)

Christmas Eve: Dysfunctional Family

- Initial state
  - Hungry(children)
  - Want(children,gifts)

Eat(person)	
Precond: Hungry(person)	Add: Full(person)
	Delete: Hungry(person)

Receive(person,object)	
Precond: Want(person, object)	Add: Have(person,object)
	Delete: Want(person,object)

- Goal state
  - Happy(children)

Play(person,object)	
Precond: Have(person, object) Full(person)	Add: Happy(person)
	Delete:

- How do we get from initial to goal state?

Dysfunctional Family: Solutions

Eat(person)	
Precond: Hungry(person)	Add: Full(person)
	Delete: Hungry(person)

Receive(person,object)	
Precond: Want(person, object)	Add: Have(person,object)
	Delete: Want(person,object)

play(person,object)	
Precond: Have(person, object) Full(person)	Add: Happy(person)
	Delete:

- Eat(children)
- Receive(children,gifts)
- Play(children,gifts)

or

- Receive(children,gifts)
- Eat(children)
- Play(children,gifts)

- Note:
  – Unknown facts are false
  – True facts remain true
  – Planning strategy?
- In this case:
  – Order of eat/gifts doesn‘t matter
  – Order of eat/play matters
  – Parents aren‘t happy
  – Parents don‘t eat
  – Parents don‘t play
  – …??

Christmas evening (2nd try)

- Initial state
  - Hungry(children)
  - Hungry(parents)
  - Want(children,gifts)

- Goal state
  - Happy(children)
  - Happy(parents)

Christmas Eve: Musical Family

- Initial state
  – Hungry(children)
  – Hungry(parents)
  – Want(children,gifts)
- Goal state
  – Happy(children)
  – Happy(parents)

Eat(person)	
Precond: Hungry(person)	Add: Full(person)
	Delete: Hungry(person)

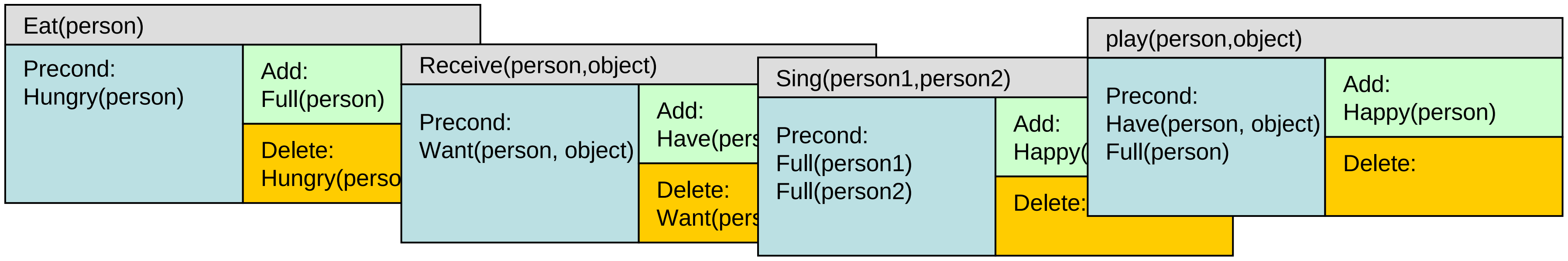
Receive(person,object)	
Precond: Want(person, object)	Add: Have(person,object)
	Delete: Want(person,object)

Sing(person1,person2)	
Precond: Full(person1) Full(person2)	Add: Happy(person2)
	Delete:

new

Play(person,object)	
Precond: Have(person, object) Full(person)	Add: Happy(person)
	Delete:

Musical Family: Solutions

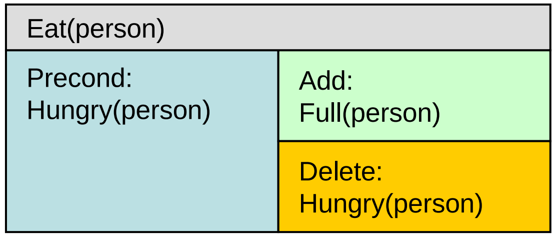


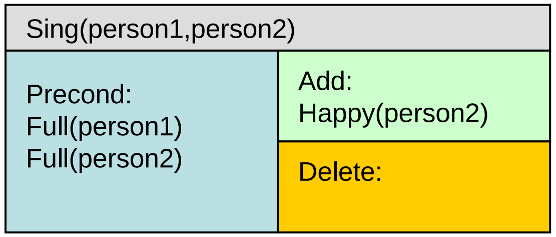
- Eat(children)
- Eat(parents)
- Sing(children,parents)
- Receive(children,gifts)
- Play(children,gifts)

- Only partial order:
  - Children can eat before or after parents
  - Singing can be done before or after gifts
  - All eat before singing

- Planning strategy?

But also this…. 8(





- Eat(children)
- Eat(parents)
- Sing(children,parents)
- Sing(parents,children)

- Not a correct model
  - Children can be happy without gifts
- Even worse:
  - …
  - Sing(parents,children)
  - Receive(parents,gifts)
  - Play(parents,gifts)

Christmas evening (3rd try)

- Initial state
  - Hungry(children)
  - Hungry(parents)
  - Want(children,gifts)
  - In(dad,Lroom)
  - In(children,Lroom)
  - Full(santa)

- Goal state
  - Happy(children)
  - Happy(parents)
  - Believe(children,santa)

Plan your own Christmas Eve

Enter(person,room)	
Precond: Outs(person,room)	Add: In(person,room)
	Delete: Outs(person,room)

Sing(person1,person2)	
Precond: Full(person1) Full(person2) Want(person1,gifts)	Add: Happy(person2)
	Delete:

Leave(person,room)	
Precond: In(person,room)	Add: Outs(person,room)
	Delete: In(person,room)

Eat(person)	
Precond: Hungry(person)	Add: Full(person)
	Delete: Hungry(person)

Eat(person)	
Precond: Hungry(person)	Add: Full(person)
	Delete: Hungry(person)

Enter(person,room)	
Precond: Outs(person,room)	Add: In(person,room)
	Delete: Outs(person,room)

Receive(person,object,pers2)	
Precond: Want(person,object) Happy(pers2)	Add: Have(person,object)
	Want(person,object) Happy(pers2)

Change(person1,person2,room)	
Precond: Outs(person1,room) Full(person1)	Outs(person2,room) Full(person2)
	Delete: Outs(person1,room)

Change(person1,person2,room)	
Precond: Outs(person1,room) Full(person1)	Outs(person2,room) Full(person2)
	Delete: Outs(person1,room)

Sing(person1,person2)	
Precond: Full(person1) Full(person2) Want(person1,gifts)	Add: Happy(person2)
	Delete:

Leave(person,room)	
Precond: In(person,room)	Add: Outs(person,room)
	Delete: In(person,room)

See(person1,person2,room)	
Precond: in(pers1,room) In(pers2,room)	Add: Believe(pers1,pers2)
	Delete:

Play(person,object)	
Precond: Have(person, object) Full(person)	Add: Happy(person)
	Delete:

STRIPS backward planning

- Choose one fact from goal state
- See how it can be achieved
  - Fact is in the add list of operator
  - No other fact from goal state is in delete list
- Remove fact from goal, add preconditions
- Iterate until all goal facts are known to be true
- Backtrack if no choice is possible

- Good choice in step 1: fewer preconditions

Christmas Eve: Family with small children

- Eat(children)
- Eat(parents)
- Sing(children,parents)
- Leave(dad,Lroom)
- Change(dad,santa)
- Enter(santa,Lroom)
- See(children,santa)
- Sing(children,santa)
- Receive(children,gifts, santa)
- Leave(santa,Lroom)
- Change(santa,dad)
- Enter(dad,Lroom)
- Play(children,gifts)

Leave(person,room)	
Precond: In(person,room)	Add: Outs(person,room)
	Delete: In(person,room)

Eat(person)	
Precond: Hungry(person)	Add: Full(person)
	Delete: Hungry(person)

Enter(person,room)	
Precond: Outs(person,room)	Add: In(person,room)
	Delete: Outs(person,room)

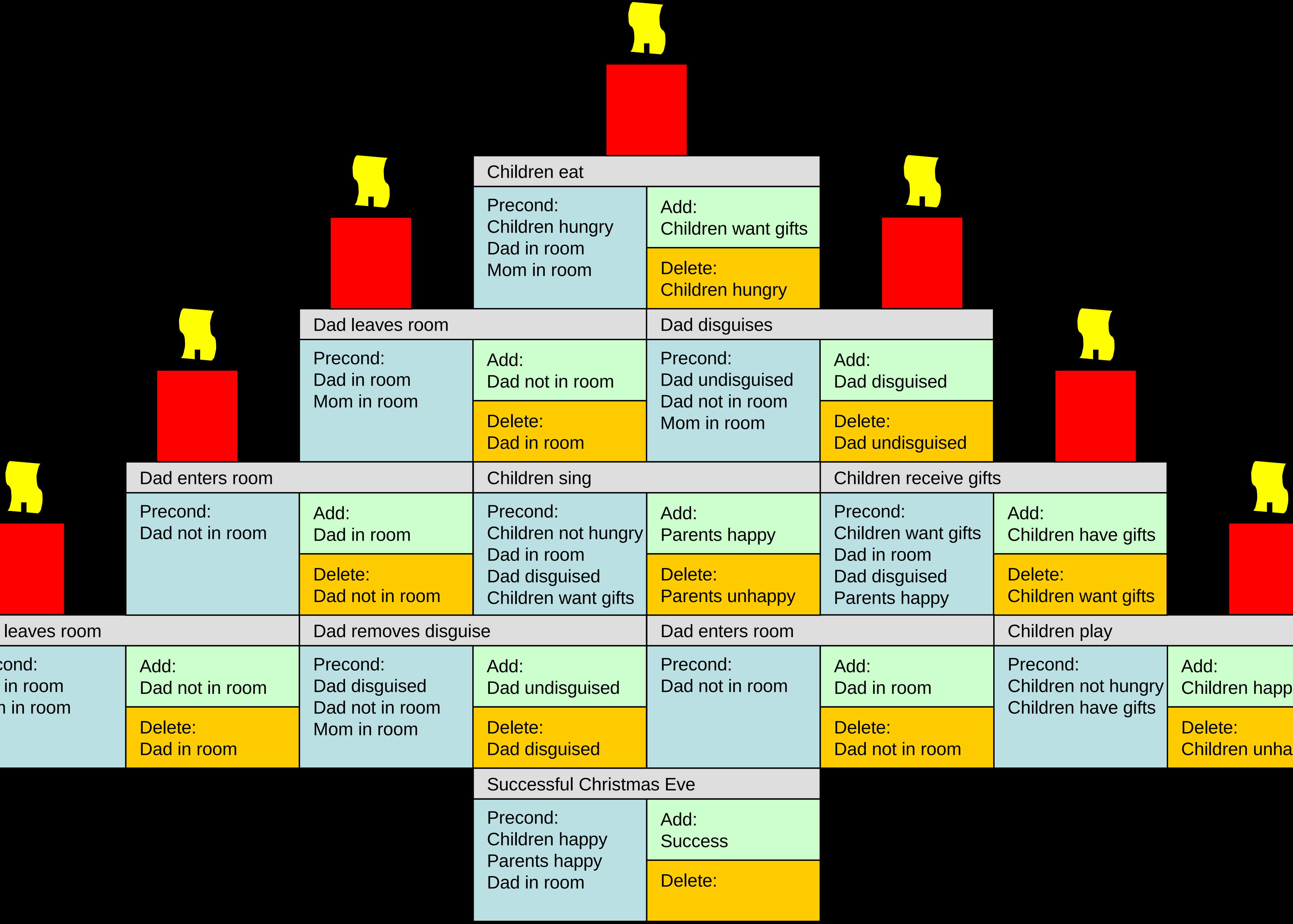
Receive(person,object,pers2)	
Precond: Want(person,object) Happy(pers2)	Add: Have(person,object)
	Want(person,object) Happy(pers2)

Change(person1,person2,room)	
Precond: Outs(person1,room) Full(person1)	Outs(person2,room) Full(person2)
	Delete: Outs(person1,room)

Sing(person1,person2)	
Precond: Full(person1) Full(person2) Want(person1,gifts)	Add: Happy(person2)
	Delete:

See(person1,person2,room)	
Precond: in(pers1,room) In(pers2,room)	Add: Believe(pers1,pers2)
	Delete:

Play(person,object)	
Precond: Have(person, object) Full(person)	Add: Happy(person)
	Delete:



Children eat	
Precond: Children hungry, Dad in room, Mom in room	Add: Children want gifts
	Delete: Children hungry

Dad leaves room	
Precond: Dad in room, Mom in room	Add: Dad not in room
	Delete: Dad in room

Dad disguises	
Precond: Dad undisguised, Dad not in room, Mom in room	Add: Dad disguised
	Delete: Dad undisguised

Dad enters room	
Precond: Dad not in room	Add: Dad in room
	Delete: Dad not in room

Children sing	
Precond: Children not hungry, Dad in room, Dad disguised, Children want gifts	Add: Parents happy
	Delete: Parents unhappy

Children receive gifts	
Precond: Children want gifts, Dad in room, Dad disguised, Parents happy	Add: Children have gifts
	Delete: Children want gifts

leaves room	
cond: in room, n in room	Add: Dad not in room
	Delete: Dad in room

Dad removes disguise	
Precond: Dad disguised, Dad not in room, Mom in room	Add: Dad undisguised
	Delete: Dad disguised

Dad enters room	
Precond: Dad not in room	Add: Dad in room
	Delete: Dad not in room

Children play	
Precond: Children not hungry, Children have gifts	Add: Children happ
	Delete: Children unha

Successful Christmas Eve	
Precond: Children happy, Parents happy, Dad in room	Add: Success
	Delete: