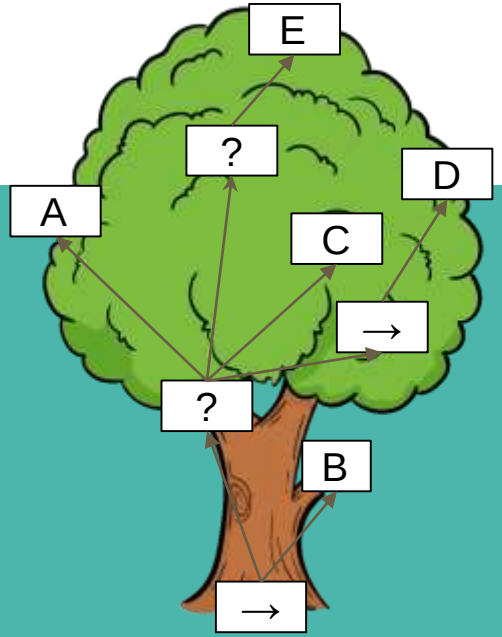
Behaviorální stromy



Behaviorální stromy (BTs)

- Stromová datová struktura umožňující měnit stavy(== *úlohy*) za běhu.
- Matematicky definované
- Alternativa STM
- Komplexní systémy

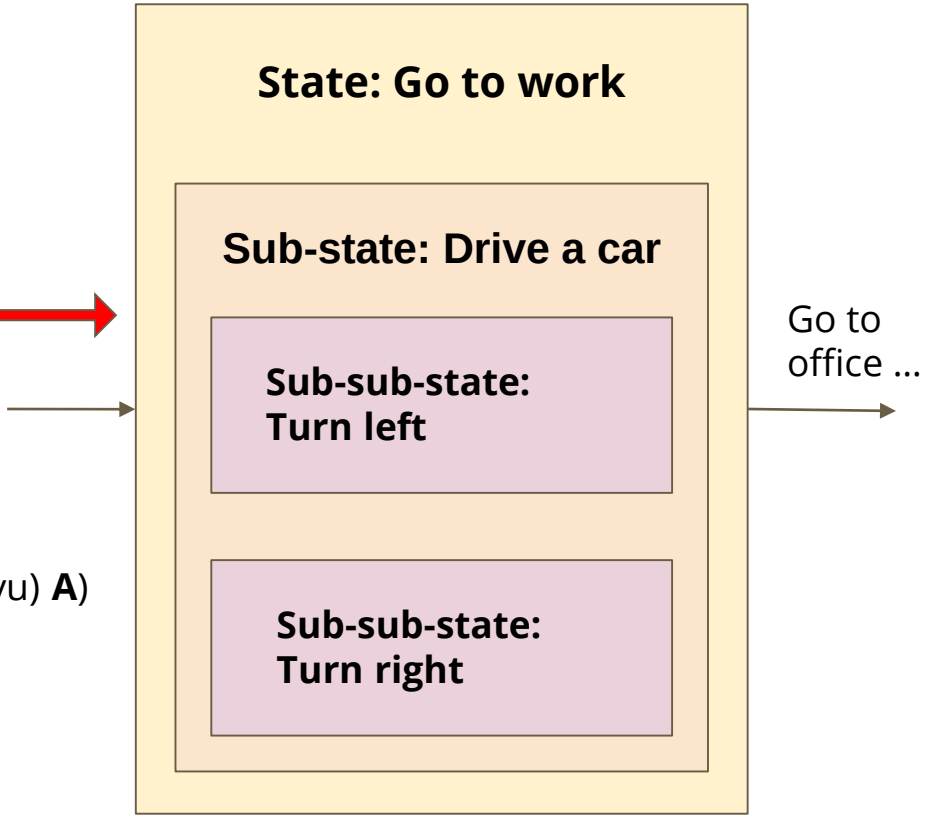
Možná definice:

„BT is a directed rooted tree where the internal nodes are called ***control flow nodes*** *and leaf nodes are called* ***execution nodes.***“

(převzato z: Behavior Trees in Robotics and AI)

Hlavní výhody

- Modularita
- Hierarchická struktura
- Přehledná grafická reprezentace
- Explicitní přechody mezi stavy
  - Sequence
    (kam pokračovat po dokončení úlohy(stavu) **A**)
  - Fallbacks
    (co dělat, když úloha(stav) **A** selže)
  - Interruptions
    (co dělat, při neočekávaném přerušení současné úlohy(stavu) **A**)



Výpočet

- Strom procházíme pomocí DFS od kořene.
- To jak DFS postupně navštěvuje jednotlivé vrcholy nazýváme *ticks*.
- Výpočet uzlu se provede ⇔ k němu dorazí *tick*.

Při výpočtu každý uzel vrací(při vynořování z rekurze) jednu z následujících tří možností:

Success | **Failure** | **Running** .

Na základě **typu (vnitřního)uzlu** do kterého se rekurze vrátí **a vráceného signálu** se rozhodne jak bude DFS pokračovat.

Uzly stromu

Control flow nodes

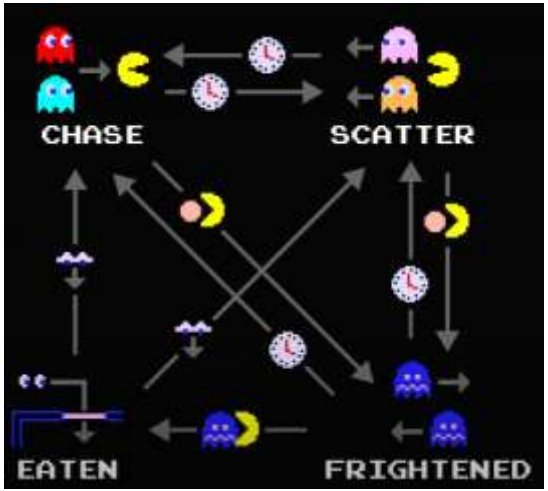
- Vnitřní uzly stromu. (=> Každý *control flow node* má vždy alespoň jednoho potomka.)
- Určují jak je strom vyhodnocen.
- Čtyři typy: ***Sequence, Fallback,*** *Parallel, Decorator*

Execution nodes

- Listy stromu.
- Obsahují konkrétní implementaci logiky aplikace.
- Dva typy: *Action, Condition*

Execution nodes

- Jsou v listech.



Action

Implementace konkrétní úlohy k vykonání.

Vrací **Success** V **Failure** V **Running**

Action

Condition

Implementace specifické podmínky,
podle níž se následně určuje „*co se stane dál*“.

=> Vrací pouze **Success** V **Failure**

Condition

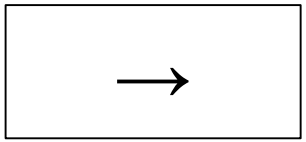
Používají se k implementaci interrupts.

Control flow nodes

Sequence

- Slouží k připojení jednotlivých *(pod)úloh* do „série“.
- Vrací **Success** ⇔ všechny jeho potomci vrací **Success**
- Vrací **Failure** v **Running** ⇔ alespoň jeden potomek vrací **Failure** v **Running**

=> **Chová se jako konjunkce!**



Sémantická značka

Algorithm 1: Pseudocode of a Sequence node with N children

```
1 for i ← 1 to N do
2     childStatus ← Tick (child(i))
3     if childStatus = Running then
4         return Running
5     else if childStatus = Failure then
6         return Failure
7 return Success
```

Control flow nodes

Fallback

- Slouží k připojení jednotlivých *úloh* „paralelně“.
- Vrací **Failure** ⇔ všechny jeho potomci vrací **Failure**
- Vrací **Success** ∨ **Running** ⇔ alespoň jeden potomek vrací **Success** ∨ **Running**

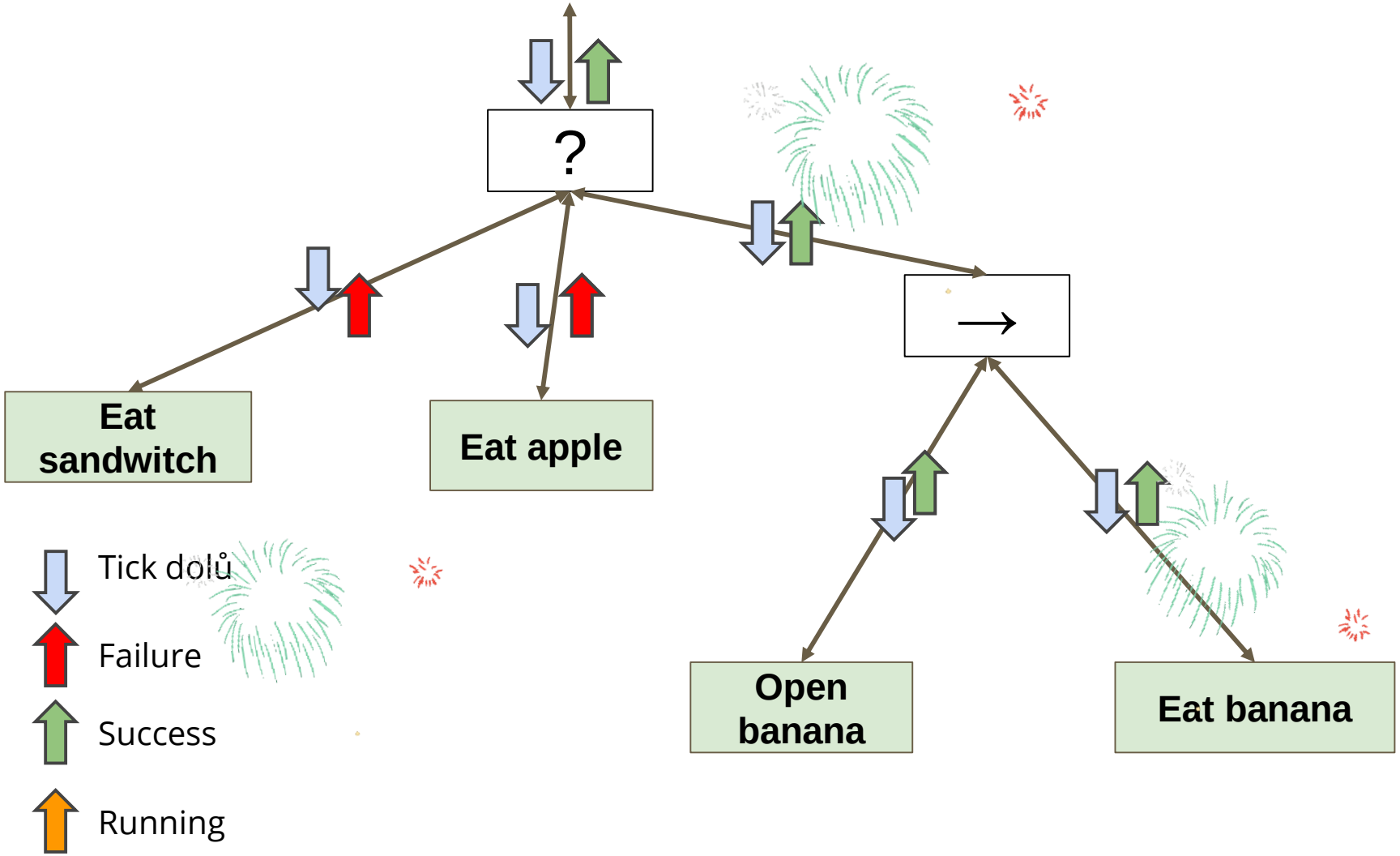
=> **Chová se jako disjunkce!**

Sémantická značka

Algorithm 2: Pseudocode of a Fallback node with N children

```
1 for i ← 1 to N do
2     childStatus ← Tick(child(i))
3     if childStatus = Running then
4         return Running
5     else if childStatus = Success then
6         return Success
7 return Failure
```

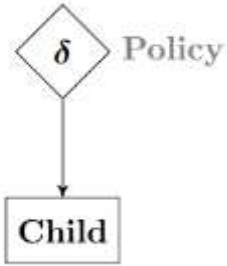
Příklad


?
→
Eat sandwitch
Eat apple
Open banana
Eat banana
Tick dolů
Failure
Success
Running

Control flow nodes – other

Decorator

- Změna navrácené hodnoty (např. negace).
- Může sloužit jako časovač (signál propustí dál až po k-tém *ticku*)
- …



Parallel

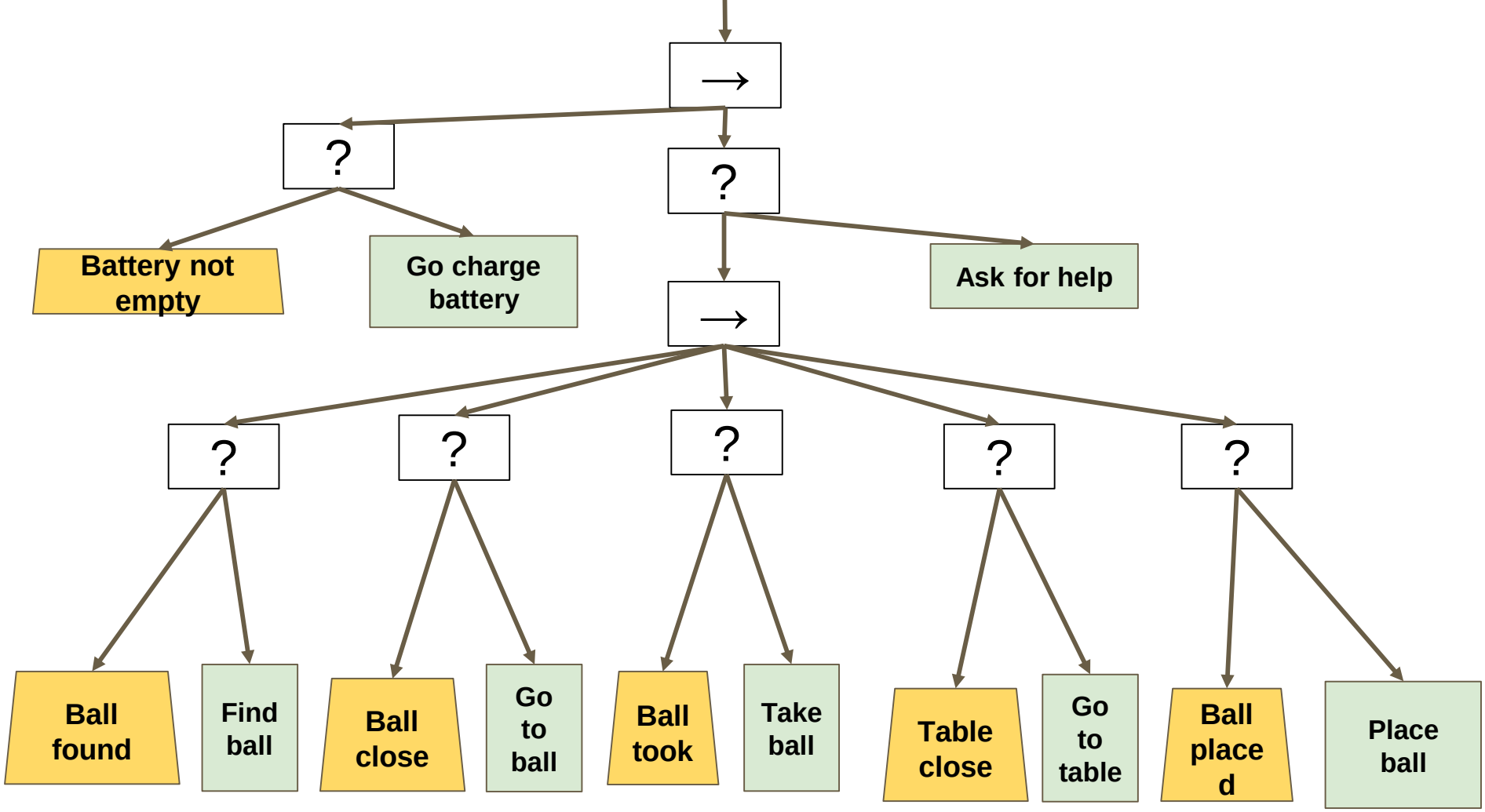
- **Success** – **M** potomků **Success**
- **Failure** – **N - M + 1** potomků **Failure**

  jinak **Running**

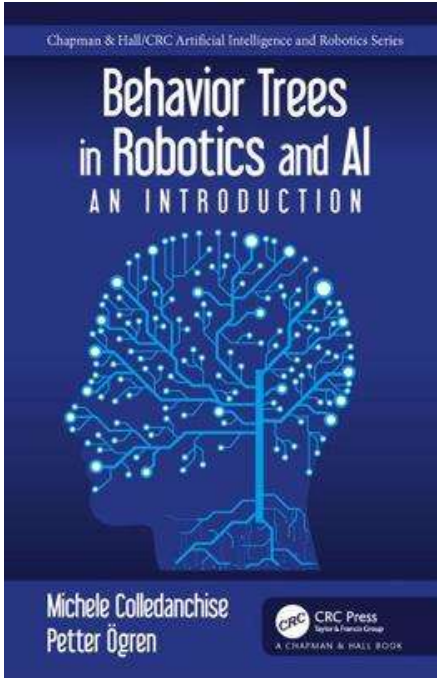
Algorithm 3: Pseudocode of a Parallel node with N children and success threshold M

```
for i ← 1 to N do
    childStatus(i) ← Tick (child(i))
if Σ_{i:childStatus(i)=Success} 1 ≥ M then
    return Success
else if Σ_{i:childStatus(i)=Failure} 1 > N − M then
    return Failure
return Running
```

Komplikovanější příklad


→
?
?
Battery not empty
Go charge battery
Ask for help
→
?
?
?
?
?
Ball found
Find ball
Ball close
Go to ball
Ball took
Take ball
Table close
Go to table
Ball place d
Place ball

Další zdroj informací



Zdroje

State pattern:

GAMMA, Erich. *Design patterns: elements of reusable object-oriented software*. Boston: Addison-Wesley, 1995. ISBN 978-0201633610.

https://refactoring.guru/design-patterns/state

https://sourcemaking.com/design_patterns/state

https://en.wikipedia.org/wiki/State_pattern

Behaviour trees:

COLLEDANCHISE, Michele a Petter ÖGREN. *Behavior Trees in Robotics and AI: An introduction*. 1. Taylor & Francis, 2018. ISBN 978-1138593732

Bc. Xeniya Valentova. *Simulace inteligentního chování zvířat v dynamickém herním prostředí*. Praha, 2016. Diplomová práce. České vysoké učení technické, Fakulta informačních technologií. Vedoucí práce Ing. Marek Žehra