DEPARTEMEN TEKNIK INFORMATIKA

REINFORCEMENT LEARNING

Kecerdasan Komputasional

Artificial Intelligence

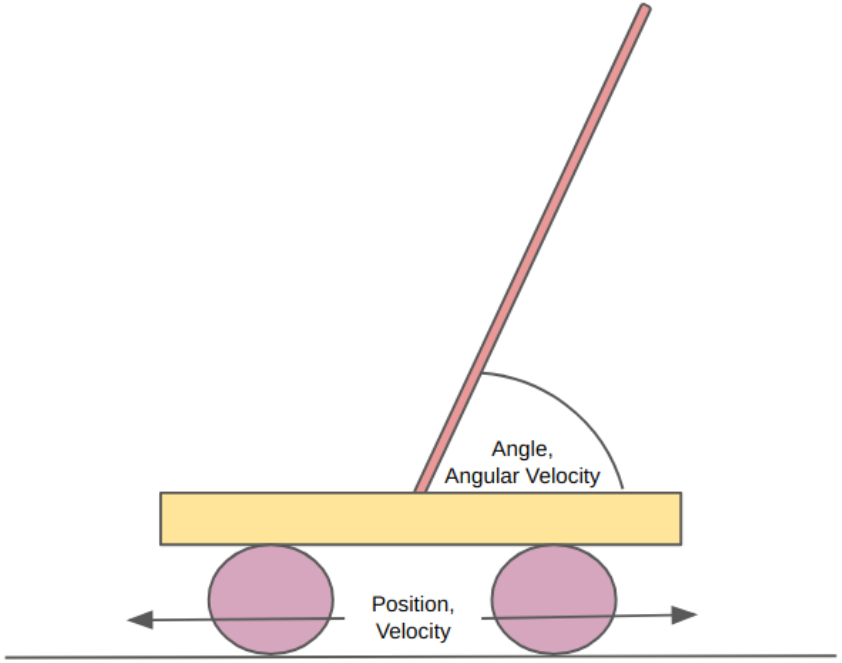
Machine Learning

- Supervised
- Unsupervised
- Semi-Supervised
- Reinforcement

Konsep *Reinforcement Learning*

- Konsep pembelajaran yang melibatkan interaksi ***Agent*** dengan lingkungan (***environment***) untuk mencapai tujuan (***goal***)
- **Agent**: mempunyai tugas untuk mencapai tujuan (***goal***)
- **Environment**: memberikan umpan balik terhadap aksi yang dilakukan Agen
- **State** (*s*) merupakan kondisi atau situasi saat ini berdasarkan persepsi Agen
- **Goal:** memilih **aksi** yang memaksimalkan ***reward***
- **Reward** (*r*) merupakan sebuah nilai untuk mengukur keberhasilan aksi dari Agen
- **Action** (*a*) merupakan aksi yang akan dipilih Agen untuk mencapai tujuan

Contoh Reinforcement Learning

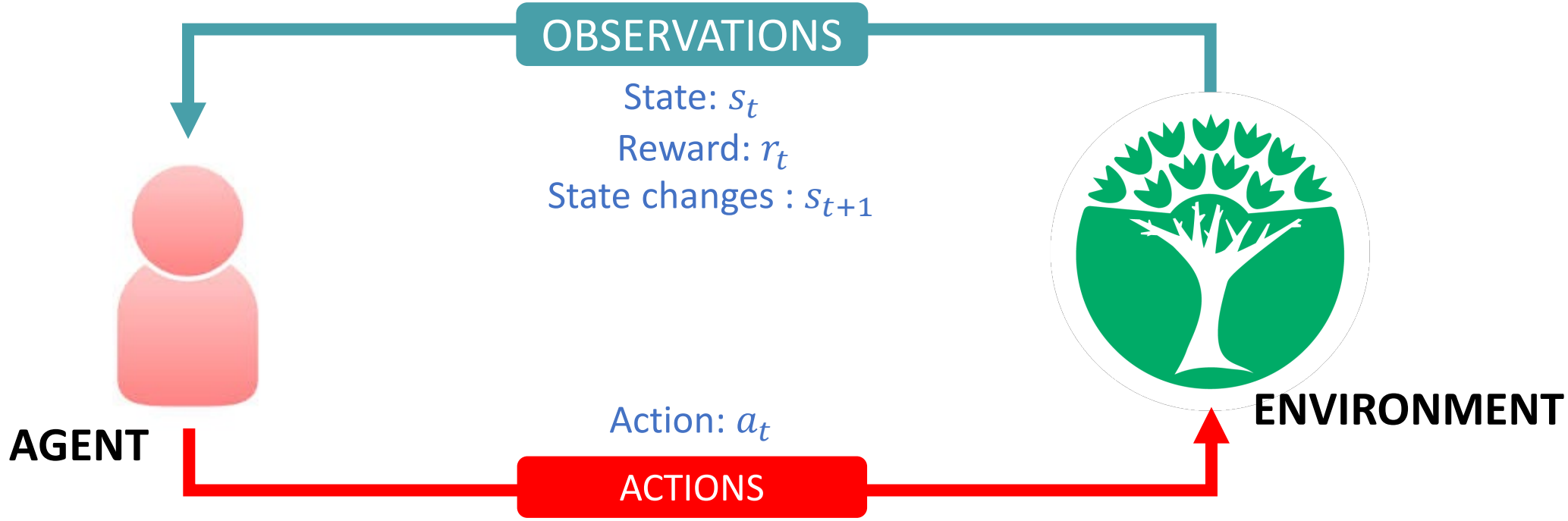


- **Cart-Pole Balancing**
  - **Goal** - Menyeimbangkan tiang diatas gerobak agar tetap berdiri
  - **State** - Sudut tiang, kecepatan sudut, posisi gerobak, kecepatan horisontal
  - **Actions** - Gaya horisontal ke gerobak
  - **Reward** – setiap step bernilai 1 jika tiang berdiri tegak

Contoh Reinforcement Learning

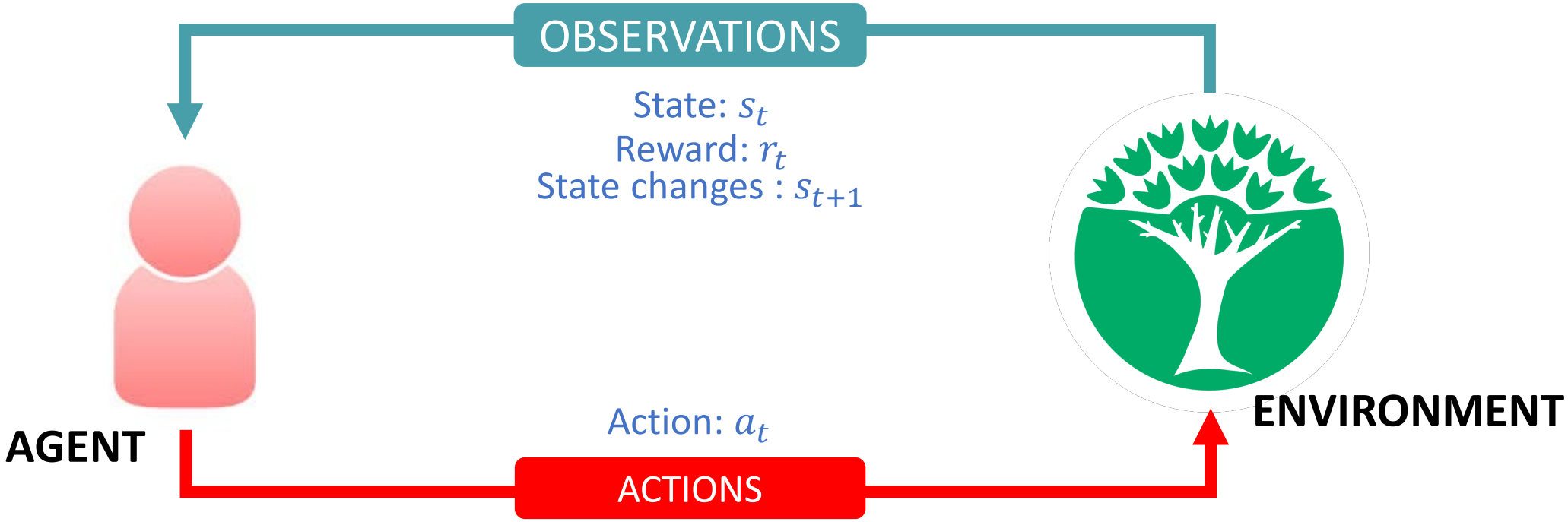
- **Bin Packing**
  - **Goal** - Mengambil barang dalam kotak dan meletakkan ke kontainer
  - **State** - Piksel-piksel pada gambar yang tertangkap kamera
  - **Actions** – Aksi-aksi yang dilakukan robot, misalnya mengambil atau meletakkan barang
  - **Reward** – Bernilai positif jika berhasil menempatkan barang dan bernilai negatif jika sebaliknya

Konsep *Reinforcement Learning*



- s_t : state pada waktu *t*
- a_t : aksi pada waktu *t*
- r_t : reward pada waktu *t*

Konsep *Reinforcement Learning*



Total Reward

$$R_t = \sum_{i=t}^{\infty} r_i = r_t + r_{t+1} + \dots + r_{t+n} + \dots$$

Discounted Total Reward

$$R_t = \sum_{i=t}^{\infty} \gamma^i r_i = \gamma^t r_t + \gamma^{t+1} r_{t+1} + \dots + \gamma^{t+n} r_{t+n} + \dots$$

γ: discount factors, $0 < \gamma < 1$

Definisi *Q-function*

Total Reward, R_t , total semua reward dengan diskon dari waktu ***t***

$$R_t = r_t + \gamma^1 r_{t+1} + \gamma^2 r_{t+2} + \ldots$$

γ: discount factors, $0 < \gamma < 1$

Q-function menangkap ***expected total feature reward*** Agen pada state ***s,*** yang melakukan aksi ***a*** tertentu

$$Q(s_t, a_t) = \mathrm{E}[R_t | s_t, a_t]$$

Bagaimana cara Agen memilih aksi dari Q-function?

$$Q(s_t, a_t) = \mathrm{E}[R_t | s_t, a_t]$$

Q-function

Agen memerlukan suatu ***policy*** $\pi(s)$ untuk memilih **aksi terbaik** pada state ***s***

Strategi: memilih aksi yang memaksimalkan ***future reward***

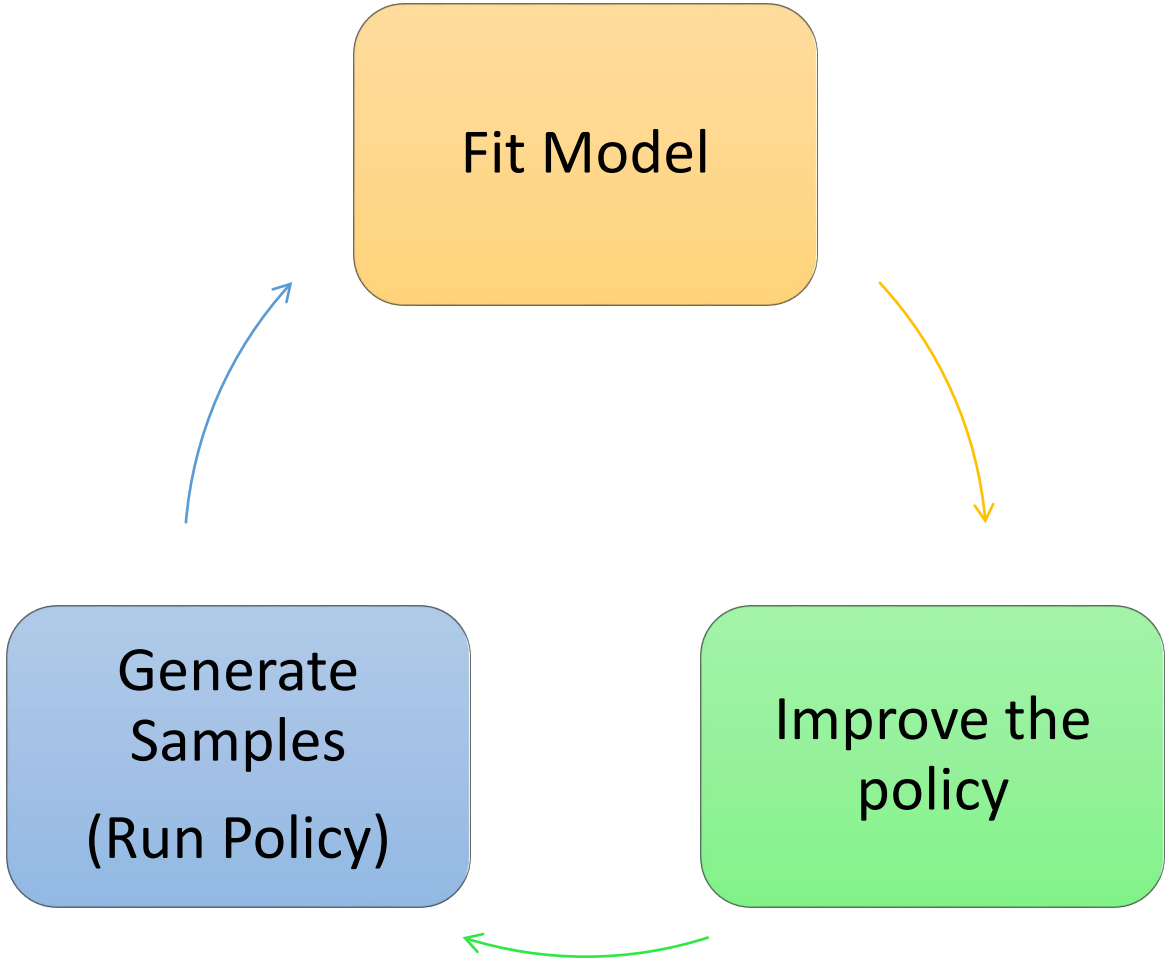
$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a)$$

π^* **optimal *policy* π**

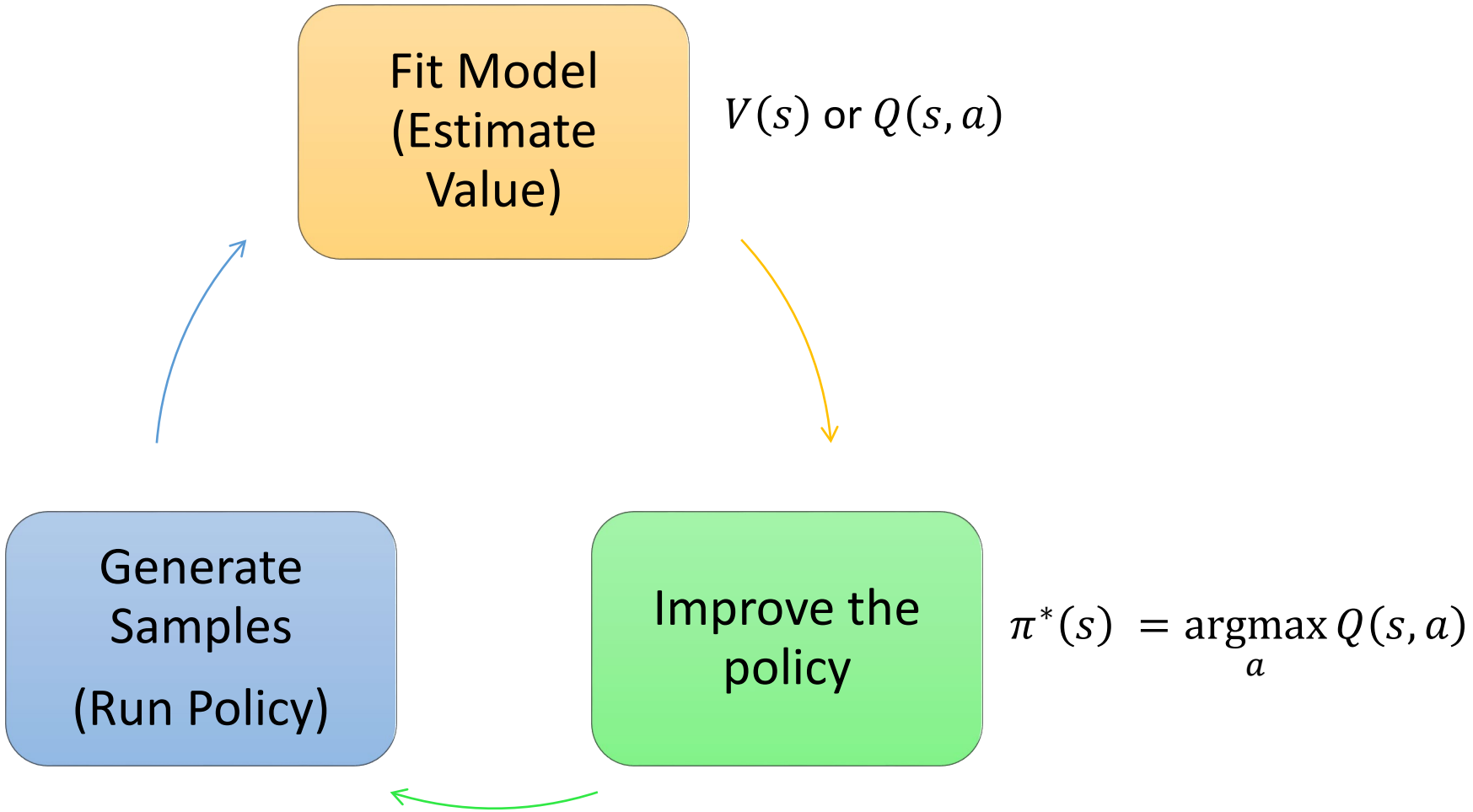
Pendekatan *Reinforcement Learning*

- Policy-based RL
  - Memilih secara langsung *optimal policy* π^*
  - Memilih *policy* yang mencapai *maximum future reward*
- Value-based RL
  - Mengestimasi *optimal value function* $Q^*(s,a)$
  - Maximum value yang tercapai oleh *policy*
- Model-based RL
  - Membangun *transition model* pada sebuah *environment*
  - Perencanaan berbasis model

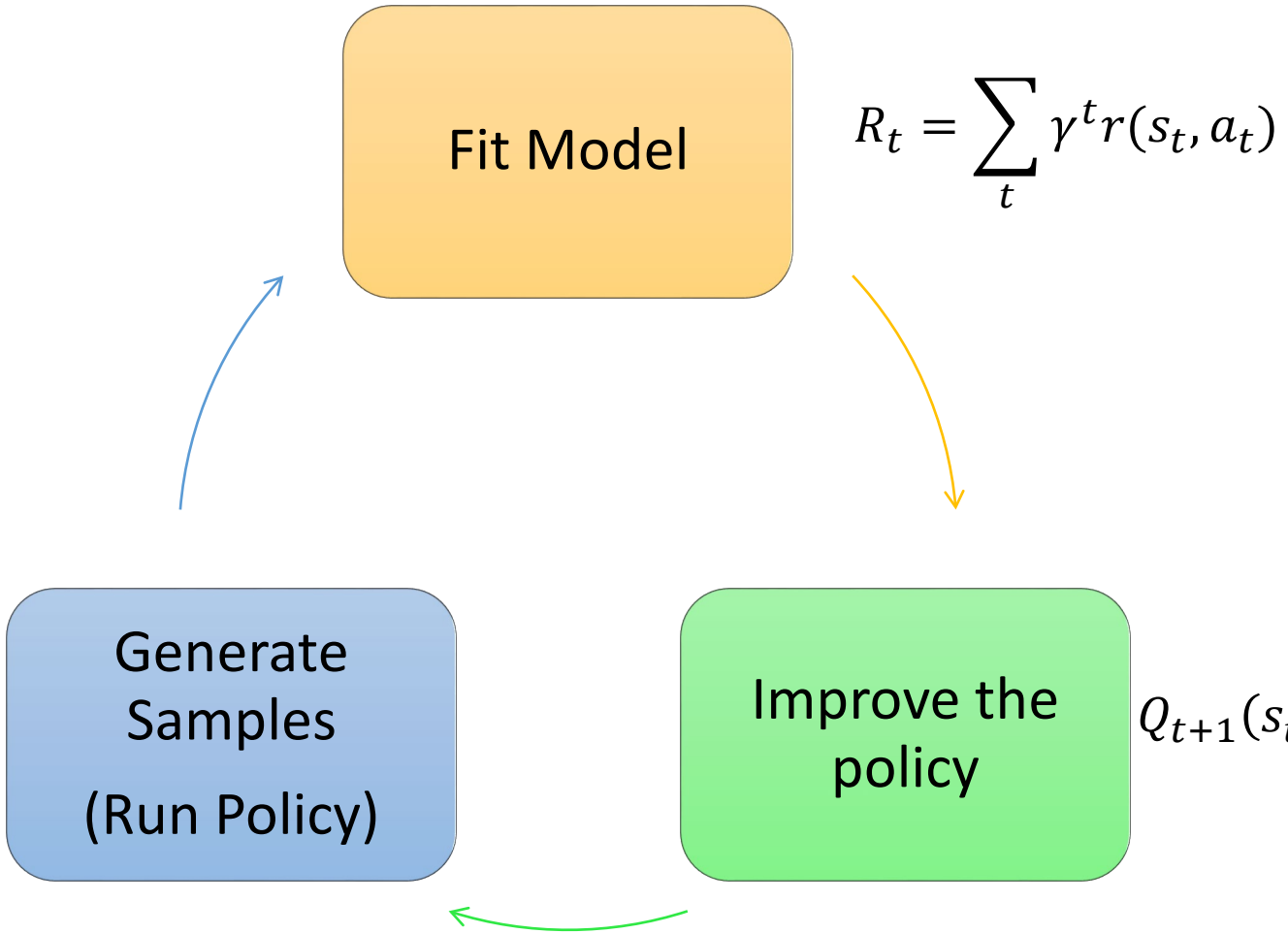
Desain Algoritma Reinforcement Learning



Policy-based Reinforcement Learning



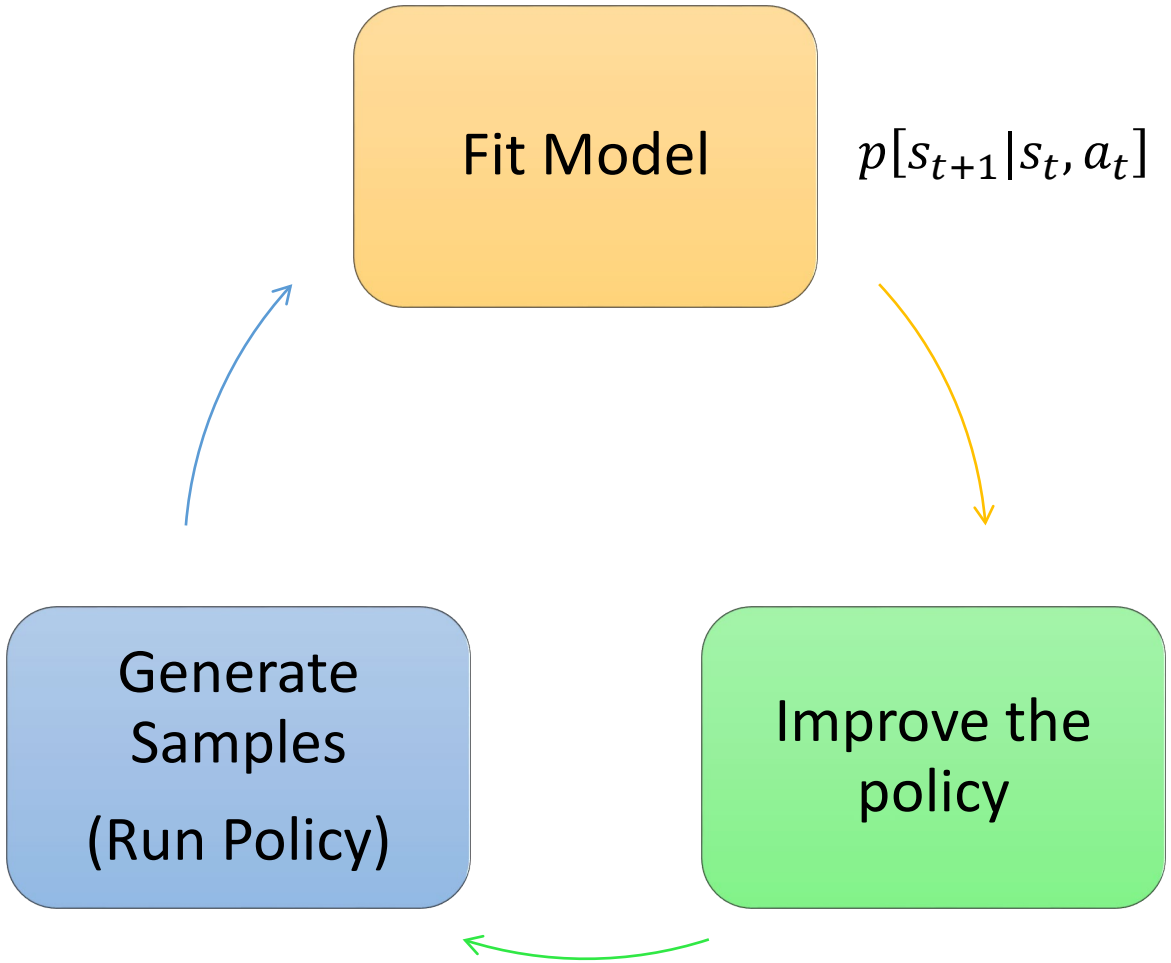
Value-based Reinforcement Learning



$$R_t = \sum_t \gamma^t r(s_t, a_t)$$

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \nabla_Q \mathrm{E}[R_t | s_t, a_t]$$

Model-based Reinforcement Learning



a. Menggunakan model (no policy)
   - Monte Carlo tree search

b. Backpropagate gradients policy

c. Menggunakan model untuk pembelajaran value function
   - Dynamic programming

Q-Learning

Menggunakan ***policy*** untuk mengestimasi *Q* yang memaksimalkan ***future reward***:

- Aproksimasi *Q** (persamaan *Bellman optimality*)
- Update setiap pasangan *(s, a)*

Learning rate

Discount factor

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \propto \left(R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\right)$$

New state

Old state

Reward

Q-Learning: Value Iteration

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \propto \left(R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)\right)$$

Learning rate

Discount factor

New state

Old state

Reward

Q-Table

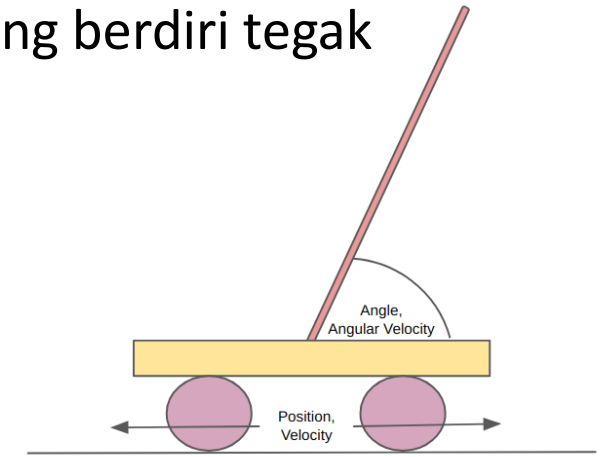
	A1	A2	A3	A4
S1	+1	+2	-1	0
S2	+2	0	+1	-2
S3	-1	+1	0	-2
S4	-2	0	+1	+1

```
initialize Q[num_states,num_actions] arbitrarily
observe initial state s
repeat
      select and carry out an action a
      observe reward r and new state s'
      Q[s,a] = Q[s,a] + α(r + γ max_a' Q[s',a'] - Q[s,a])
      s = s'
until terminated
```

Sumber: Lex Fridman, Deep Reinforcement Learning, MIT Course 2018 : Introduction to Deep Learning

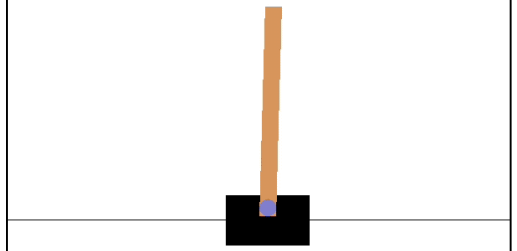
Contoh Penerapan Q-Learning

- **Cart-Pole Balancing**
  - **Goal** - Menyeimbangkan tiang diatas gerobak agar tetap berdiri
  - **State** - Sudut tiang, kecepatan sudut, posisi gerobak, kecepatan horisontal
  - **Actions** - Gaya horisontal ke gerobak
  - **Reward** - setiap step bernilai 1 jika tiang berdiri tegak

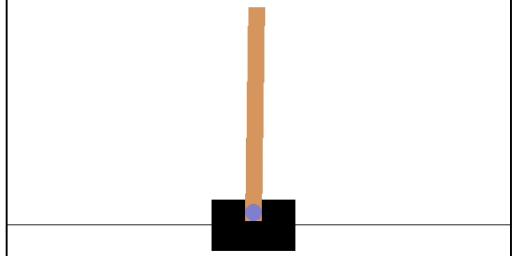



```
import gym
env = gym.make('CartPole-v0')
for i_episode in range(20):
    observation = env.reset()
    for t in range(100):
        env.render()
        print(observation)
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
        if done:
            print("Episode finished after {} timesteps".format(t+1))
            break
env.close()
```

Tanpa Q-Learning



Dengan Q-Learning



Kelemahan Q-Learning

Value Iteration tidak praktis:

- State atau action yang terbatas
- Tidak dapat mengeneralisir state yang belum diketahui

Contoh **Breakout** game

State: screen pixels

- Ukuran gambar: 84×84(resized)
- Consecutive **4** images
- Grayscale with **256** gray level

$256^{84 \times 84 \times 4}$ rows pada Q-table!

Solusi pendekatan Deep Q Learning!

- TERIMA KASIH -