THE UNIVERSITY OF TEXAS AT AUSTIN

Probabilistic Control

John Thywissen

C S 393R, Fall 2009, Prof. P. Stone

Final Project • 1 Dec 2009

Genesis: Previous projects’ Behavior code

```
 void KickerBehavior::processEvent(const EventBase& event) {
switch (event.getGeneratorID()) {
case EventBase::visObjEGID: {
    updateState(dynamic_cast<const VisionObjectEvent&> (event));
    switch (currentState) {
    case ballScan: {
        setModeLedRgb(1.0, 0.0, 0.0);
        if (ball_is_visible) {
            out(fsmState, "\tBall found, transitioning to moveToBall.");
            currentState = moveToBall;
            MMAccessor<WalkMC> (walker_id)->setTargetVelocity(0.0, 0.0, 0.0);
        }
        break;
    }
    case moveToBall: {
        setModeLedRgb(1.0, 1.0, 0.0);
        if (!ball_is_visible) {
            if (ballLostFrameCount >= ballLostFrameThreshold) {
                out(fsmState, "\tBall lost, transitioning to ballScan.");
                MMAccessor<WalkMC> (walker_id)->setTargetVelocity(0, 0, 0);
                erouter->addTimer(this, scan_timerID, scan_timeout, true);
                currentState = ballScan;
                break;
            } else {
                ballLostFrameCount++;
                break;
            }
        }
        ballLostFrameCount = 0;
        gazeAtBall(middle);
        float nextForward = distanceController.getNext(ball_distance, ball_target_distance);
        float ballTurnAngle = ball_horiz_angle + head_pan_angle;
        float nextTurn = turnController.getNext(0.0, ballTurnAngle);
        out(calculation, "nextForward   = ", nextForward);
        out(calculation, "ballTurnAngle = ", ballTurnAngle);
        out(calculation, "nextTurn      = ", nextTurn);
        if (fabs(ball_distance - ball_target_distance) >= 1.0 || fabs(ballTurnAngle) >= 0.4) {
            MMAccessor<WalkMC> (walker_id)->setTargetVelocity(nextForward, 0.0, nextTurn);
        } else {
            out(fsmState, "\tArrived at ball, transitioning to goalScan.");
```

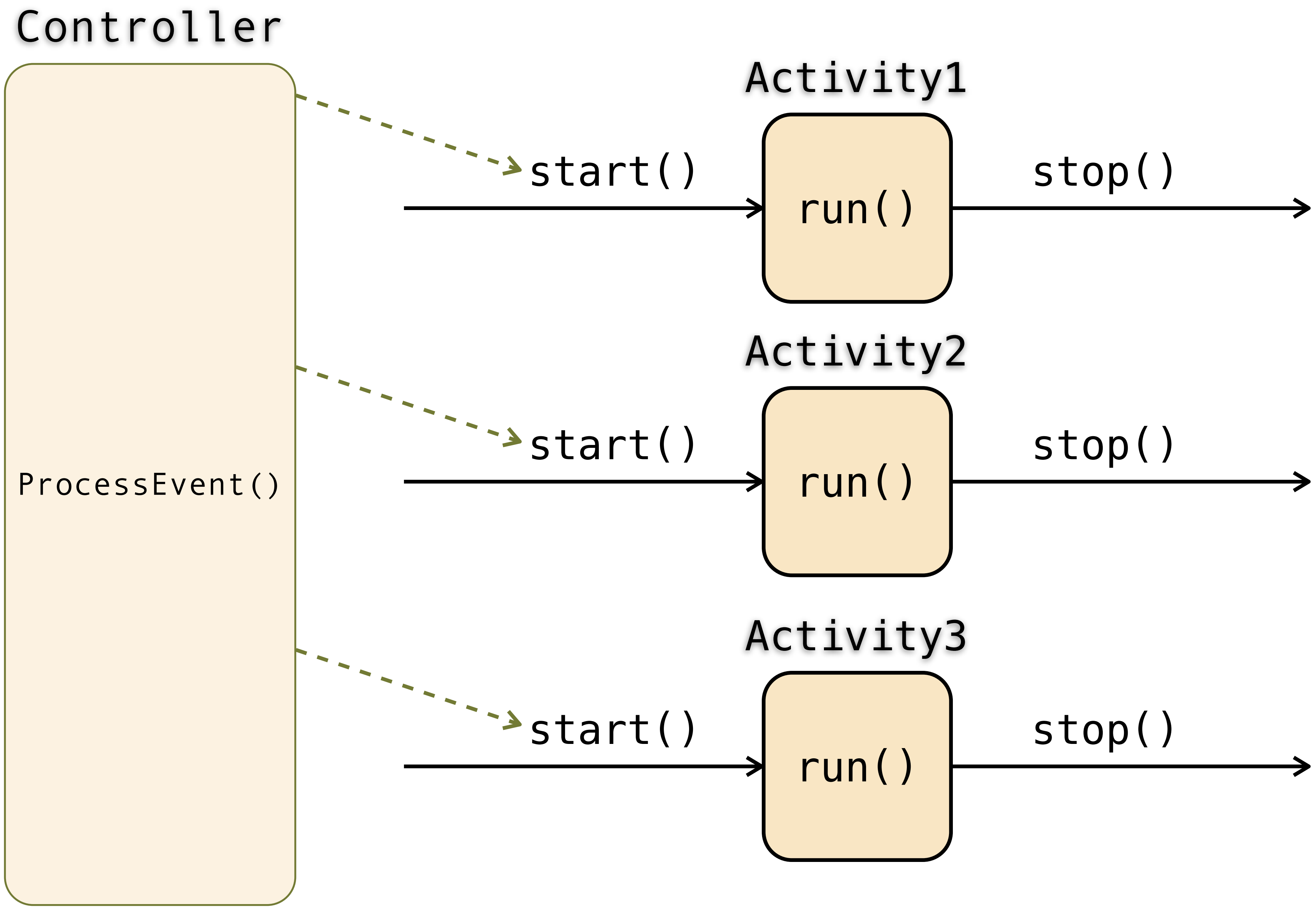
⇐ big, hairy switch/case

⇐ transition duplicated in mult. places

⇐ steady state code mixed with transition

⇓⇓ 7 pages of spaghetti... ⇓⇓

1: An easy-to-use Finite State Machine controller, for better modularity and debugging



1: An easy-to-use Finite State Machine controller, for better modularity and debugging

```
  void KickerBehavior::ballScan::start() {
    setModeLedRgb(1.0, 0.0, 0.0);
    erouter->addTimer(this, scan_timerID, scan_timeout, true);
}
```

⇐ simple, clean methods

```
void KickerBehavior::ballScan::stop() {
    MMAccessor<WalkMC> (walker_id)->setTargetVelocity(0.0, 0.0, 0.0);
}
```

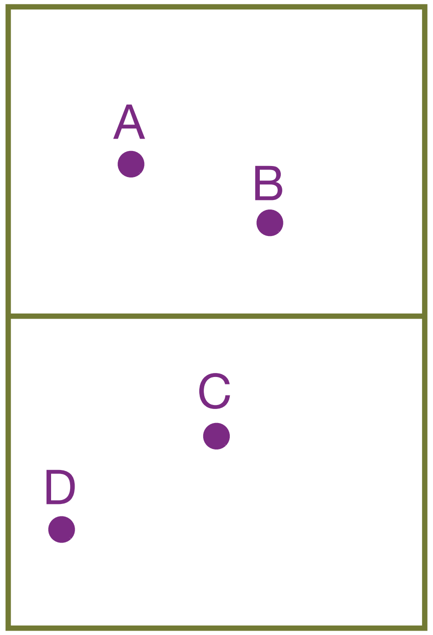
⇐ transition code in one place, near steady state code

```
void KickerBehavior::ballScan::run(const EventBase& event) {
    switch (event.getGeneratorID()) {
    case EventBase::timerEGID: {
          float scanAngle = head_pan_angle + pan_inc;
          if (scanAngle > pan_max) {
               MMAccessor<HeadPointerMC> (headpointer_id)->defaultMaxSpeed(1);
               scanAngle -= (pan_max - pan_min);
          } else {
               MMAccessor<HeadPointerMC> (headpointer_id)->defaultMaxSpeed(.1);
          }
          MMAccessor<HeadPointerMC> (headpointer_id)->setJoints(head_tilt_default, scanAngle, head_nod_default);
    }
}


void KickerBehavior::moveToBall::start() {
    setModeLedRgb(1.0, 1.0, 0.0);
}
```

⇐ activities modularized, not interleaved

```
void KickerBehavior::moveToBall::stop() {
    MMAccessor<WalkMC> (walker_id)->setTargetVelocity(0.0, 0.0, 0.0);
}

void KickerBehavior::ballScan::run(const EventRecord& event) {
    switch (event.getGeneratorID()) {
          case EventBase::visObjEGID: {
          gazeAtBall(middle);
          float nextForward = distanceController.getNext(ball_distance, ball_target_distance);
          float ballTurnAngle = ball_horiz_angle + head_pan_angle;
          float nextTurn = turnController.getNext(0.0, ballTurnAngle);
    }
```

2: Dempster-Shafer belief functions for world state belief, used to trigger state transitions



Bayesian

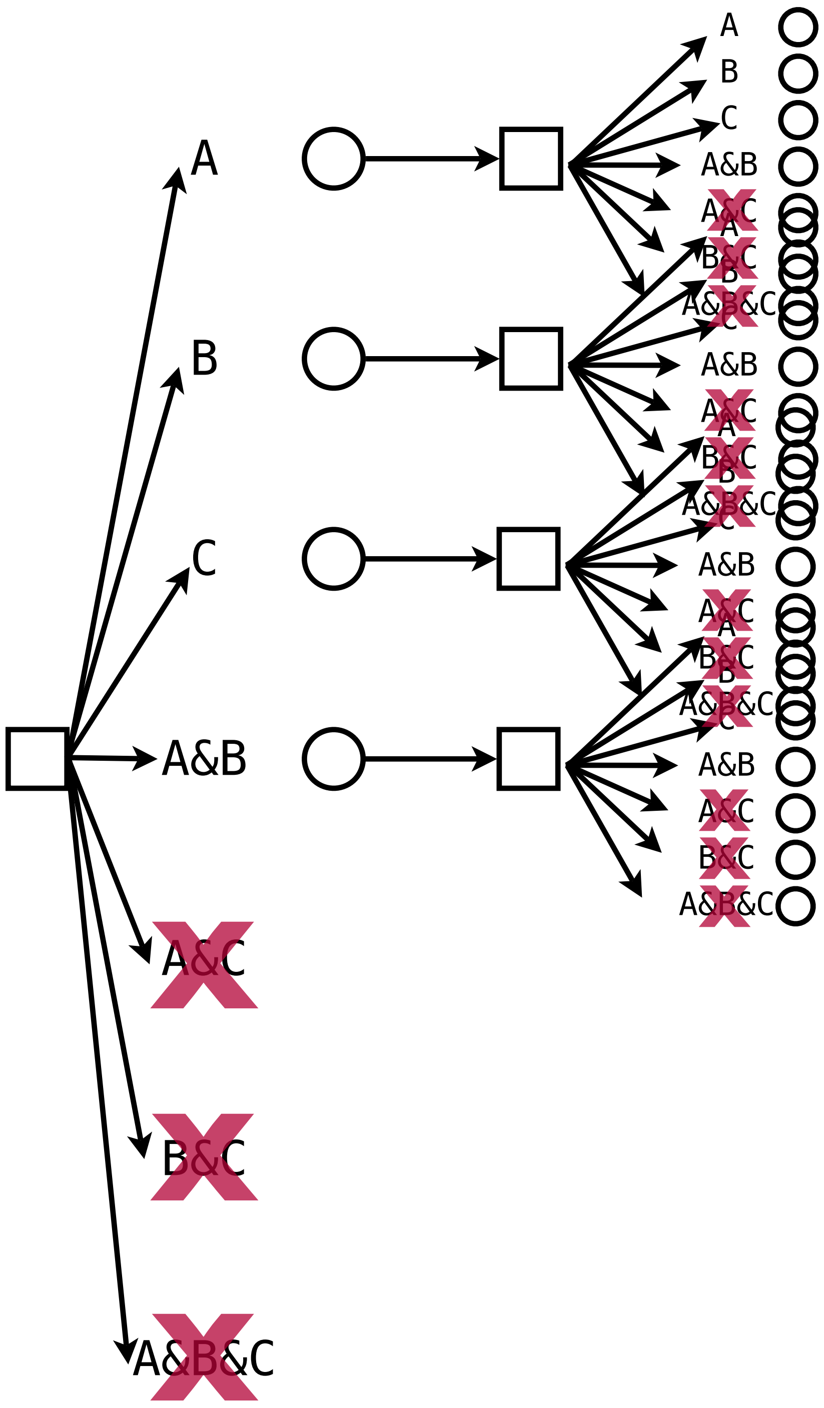
	A	B	C	D
P=	0.25	0.25	0.25	0.25

Dempster-Shafer

		A	B	C	D	A∨B	A∨B∨C∨D
	m=	0.0	0.0	0.0	0.0	0.8	0.2
Is it certainly true?	Bel=	0.0	0.0	0.0	0.0	0.8	1.0
Could it be true?	Pl=	0.8	0.8	0.2	0.2	1.0	1.0

```
...kickBall::preconditionsMet(...) {
   return beliefState[ballLoc].bel(inFrontOfUs);
}

...kickBall::preconditionThreshold(...) {
   return 0.7F;
}
```

3: Multiple actions activated to max. objective function, subject to action compatibility



```
float ...::inherentStateValue(...) {
   float belBallInGoal = beliefState[ballLoc].bel(inGoal);
   return 10.0F +
       belBallInGoal * belBallInGoal * 1250.0F;
}
```

```
float ...::walkToBall::cost(...) {
   return 2.0F;
}
```

```
bool ProbBehavTest1::isFeasibleActivitySet(const
ActivitySet& checkActivitySet) {
   // we cannot kick and look or walk simultaneously
   return !((checkActivitySet.count(&lookForBall) > 0
               || checkActivitySet.count(&walkToBall) > 0)
              && checkActivitySet.count(&kickBall) > 0);
}
```

1: An easy-to-use Finite State Machine controller, for better modularity and debugging

2: Dempster-Shafer belief functions for world state belief, used to trigger state transitions

3: Multiple actions activated to max. objective function, subject to action compatibility

Don’t collapse your nice probabilistic world state belief with clunky deterministic behavior planning!