Examples of Action Failure

We are working with actions that are specified as $\{Pre\}a\{Post\}(d)$ where Pre is a set of literals (indicating a conjunction of preconditions) a is a term (the action), $Post$ is a set of literals (indicating a conjunction of post conditions) and d is an expected duration.

In theory this states that if Pre hold and a is executed then after d time the action will have completed and $Post$ will hold. Actions can fail both by timing out and if not all post-conditions hold.

I'm interested in creating some kind of taxonomy of failures with an idea of what a new action spec would look like, the implications of this kind of failure on Rafael's reconfigurability framework, and with an at least vaguely plausible example of each kind.

Taxonomy follows, but not necessary in a sensible order.

1 Action no longer has a result

General Description When action a is executed and pre-conditions Pre hold then it terminates but $Post$ no longer holds.

Simple Abstract Example A simple abstract example is an action $\{p_1\}a\{p_2\}(1s)$. The new action spec is $\{p_1\}a\{\}(1s)$.

First thoughts on Learning In a completely static environment, learning about this failure should be simple since nothing will change in the environment when a is executed. In a dynamic environment where other things are happening then it might be necessary to observe that there is no pattern to the observations that change when a is executed and so conclude that the postconditions are false. There is obviously a case where a has just become really unreliable – i.e., sometimes p_2 still holds, but I think that can be treated as a special case of the dynamic environment problem.

Implication for Reconfigurability It's difficult to see that much can be done with this beyond removing all plans that involve this action.

Vaguely Plausible Example The "slippy gripper" is an example of this. It's action spec is something like $\{empty\}pick\_up\{gripping\}(10s)$ and the new action spec is $\{empty\}pick\_up\{\}(10s)$. The examples involving blocked routes between way points are also examples of this - for these examples we need to work on plausible recovery mechanisms after failure (which I'm not really considering here since I don't think the recovery mechanism depends upon the cause of failure but on the application – for instance the slippy gripper can probably just try again, while a rover moving between waypoints may need to perform some "figure out where I am" action on failure and then attempt some new strategy from there (e.g., call for help, or a new route, or some such)).

2 Action is partially successful

General Description When action a is executed and pre-conditions Pre hold then it terminates but while some of the literals in $Post$ hold, not all of them do.

Simple Abstract Example A simple abstract example is an action $\{p_1\}a\{p_2, p_3\}(1s)$. The new action spec is $\{p_1\}a\{p_2\}(1s)$.

First thoughts on Learning The learning considerations include those in section 1. However if p_2 sometimes also fails to happen then it will be necessary for the learning process to decide that it nevertheless is a result (still) of doing a. Possibly we need some kind of threshold or Markov chain representation representation to help us decide when something that sometimes happens is, or is not, a result of an action.

Implication for Reconfigurability Some plans may rely on both p_2 and p_3 happening and these will have to be abandoned (or a replacement action found) but any plans only relying on p_2 happening can be retained.

*Notes in this series are for ϵ-baked ideas, for $1 \geq \epsilon \geq 0$. Only exceptionally should they be cited.

Vaguely Plausible Example It's hard to think of good examples for this since good engineering practice would suggest trying to avoid actions with more than one specified outcome. One I thought of is a rover which normally sends messages from a transmitter at waypoint 3. So the specification of the action "send message" (or possibly send message from 3) is $\{\top\}send\_message\_3\{at(waypoint\_3), message\_sent\}(10m)$. If the transmitter is broken or moved then the spec becomes $\{\top\}send\_message\_3\{at(waypoint\_3)\}(10m)$ - so it might still be useful to use the action to get to way point 3. Plans that need the message sent will need to adapt to send the robot to a different transmitter. Similar examples can probably be constructed for any action which amounts to "go to X and do Y" where Y is no longer working.

3 Action has Different Outcome

General Description When action a is executed and pre-conditions Pre hold then it terminates but with different post-conditions.

Simple Abstract Example A simple abstract example is an action $\{p_1\}a\{p_2\}(1s)$. The new action spec is $\{p_1\}a\{p_3\}(1s)$.

First thoughts on Learning I think the learning issues are actually the same as in section 2. It's a matter of deciding which observed changes in the environment happen often enough to count as outcomes of the action.

Implication for Reconfigurability Some plans may have to be dropped or reconfigured to use new actions entirely, others may be adaptable with the addition of extra actions.

Vaguely Plausible Example The original sticky wheel is an example of this. But for a simpler case consider some kind of warehouse robot that follows paths on the warehouse floor to move been waypoint. There are regular marks/QR codes/some such along these paths to enable the robot to detect where it is. It's motors have deteriorated so it no longer moves as far along a path as anticipated. Suppose it has an action that will move it forward 2 markers along a path. So the old action spec is $\{at\_path1\_1\}forward\_2\{at\_path1\_3\}(10s)$ (if you are at 1 on path 1 and move forward for 2 seconds you will be at 2. The new action spec is $\{at\_path1\_1\}forward\_2\{at\_path1\_2\}(10s)$. Obviously reconfiguration can then take place just to do $forward\_2$ twice.

4 Adding variables

The case in section 3, at least, becomes more interesting (and plausible) if we throw in variables:

Suppose our warehouse robot has an action that will move it forward X markers along a path. So the old action spec is $\{on(P, X)\}forward(Y)\{on(P, X + Y)\}(10s)$ (if you are at X on P and move forward for 10 seconds you will be at $X + Y$. The new action spec is $\{on(P, X)\}forward(Y)\{on(P, \frac{(X+Y)}{2}\}(10s)$

But now the learning has to learn a relationship between P, X, Y and the output which may be more challenging – or may not since the postcondition is still the same predicate, its just the arguments that have changed so that may give more control.

5 Time Outs

Important I think we should make it clear that our time-outs indicate a point where an action should be aborted if some unknown condition for the action to cease executing has not already occurrent (that unknown condition may also be a time – for instance in all the sticky wheel variant cases but this isn't the time out (!))

I think there are time out variants on all the previous examples – i.e., when the action is aborted some new post condition has been achieved: section 1 the robot times out when its route is blocked and the action is now bascially useless; 2 the robot times out while still waiting for the message to send; and 3 the hallway example we used in the EMAS paper where the robot ends up at a different way point when it times out because of its obstacle avoidance behaviour. We should consider all of these but I think the new action learnt will be as in those cases.

The interesting variant of course is where the robot times out just because it has become slower.

General Description When action a is executed and pre-conditions Pre hold then it times out, but if allowed to run longer it would have succeeded.

Simple Abstract Example A simple abstract example is an action $\{p_1\}a\{p_2\}(1s)$. The new action spec is $\{p_1\}a\{p_2\}(2s)$.

First thoughts on Learning The learning her needs to consider not just trying the action lots of times and gathering data on outcomes but also allowing the action to run for longer and gather data on outcomes. This obviously increases the search space and it might be useful to have the learning guided in some way – e.g., the action spec could include some flag that deterioration might cause time outs to need extension. If we have variables/parameters then it may be that noticing that variable parameter values have reduced (e.g., only half the distance is covered) may be an indication that more time is needed rather than new post-conditions.

Implication for Reconfigurability If the action simply needed more time then there shouldn't be much need for reconfiguration unless timings are critical (in which case they should be explicit post-conditions).

Vaguely Plausible Example We can adapt the warehouse example so the old action spec is $\{at\_path1\_1\}forward\_2\{at\_path1\_3\}(10s)$ (if you are at 1 on path 1 and move forward for 2 seconds you will be at 2). The new action spec is $\{at\_path1\_1\}forward\_2\{at\_path1\_3\}(20s)$.

6 Action fails only for some parameters

General Description When action $a(\bar{X})$ is executed for some set of parameters, $\bar{X}$, and pre-conditions Pre hold then it terminates with failure for some $\bar{X}$.

Simple Abstract Example A simple abstract example is an action $\{p_1(X)\}a(X)\{p_2(X)\}(1s)$. Which can be called with either $X = c_1$ or $X = c_2$. It succeeds when $X = c_1$ and fails when $X = c_2$ (presumably with all the possible failure variants above but let's look at just the case in section 1). It needs to be converted to either $\{p_1(c_1)\}a(c_1)\{p_2(c_1)\}(1s)$ or $\{p_1(X), X \neq c_2\}a\{p_2(X)\}(1s)$.

First thoughts on Learning This will involve noticing that the new observations are as expected for some parameters and not for others....

Implication for Reconfigurability Some plans may have to be dropped or reconfigured to use new actions entirely, others may not need any modification if they don't require X to equal c_2.

Vaguely Plausible Example The version of the "slippy gripper" problem where the gripper can still pick up tiles, but can no longer pick up screws. So the action, $\{on\_table(X)\}pickup(X)\{holding(X)\}(2s)$ becomes $\{on\_table(tile)\}pickup(tile)\{holding(tile)\}(2s)$ (and also $\{on\_table(screw)\}pickup(screw)\{\}(2s)$). We haven't thought about action with multiple specifications to account for multiple initial conditions (and in general this can by handled by allowing disjunctions in pre-conditions – but I think we might want multiple pre-conditions since this is easier to handle in a logic programming way).

There's a variant on this example where, instead of learning that the action now only succeeds for some parameters, it needs to learn that it now only succeeds in some situation (e.g., when the hall is empty – for the hall traversal example).

7 Conclusion

This is my first attempt at a rough categorisation of action failures and the target new specs to be learned. My feeling is that we need to build a test suite of simple examples that cover all these cases before attempting the more elaborate and realistic examples that Peter has been looking at.