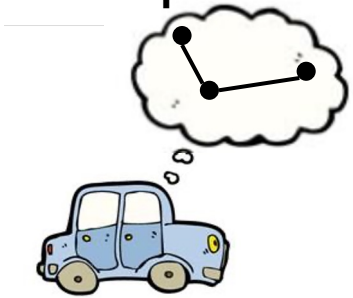
Action-Failure Resilient Solution Search

Motivations

Deterministic planning provides valid plans



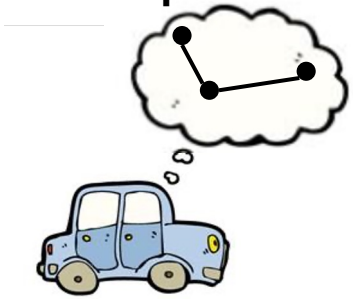
Real World

But plan execution can fail

Planning is done using **abstract models (e.g. PDDL)**

Actions can fail in *unexpected ways*
It is impossible to anticipate all possible failures

Deterministic planning provides valid plans



Planning is done using **abstract models (e.g. PDDL)**

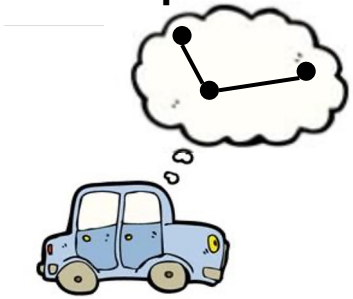
Real World

But plan execution can fail

Actions can fail in *unexpected ways*
It is impossible to anticipate all possible failures

What can we do?

Deterministic planning provides valid plans



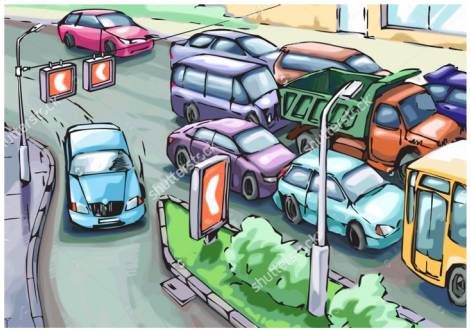
Planning is done using **abstract models (e.g. PDDL)**

Real World

But plan execution can fail

Actions can fail in *unexpected ways*
It is impossible to anticipate all possible failures

What can we do?



Find **solutions resilient** to a number of failures (with *repair guarantees* in case of action failures)

Agnostic to the particular reason of failure

Resilient Solutions

(Bounded) Resilient Solutions

A **K-Resilient problem** is a pair <Π, K> where:

- Π = Fully observable deterministic search problem
- K is an integer denoting a **number of action failures** we should be able to recover from, if they occur in a sequence of actions (a *plan*) solving Π.

K-Resilient Plan solving <Π, K> is

> A plan for Π that guarantees **we can recover from *up to K* action failures** during its execution, and still achieve the goals.

Main Assumptions and Definitions

Assumptions about the failure model: when an action fails, (1) the current state remains the same, and (2) the faulty action cannot be later reused.

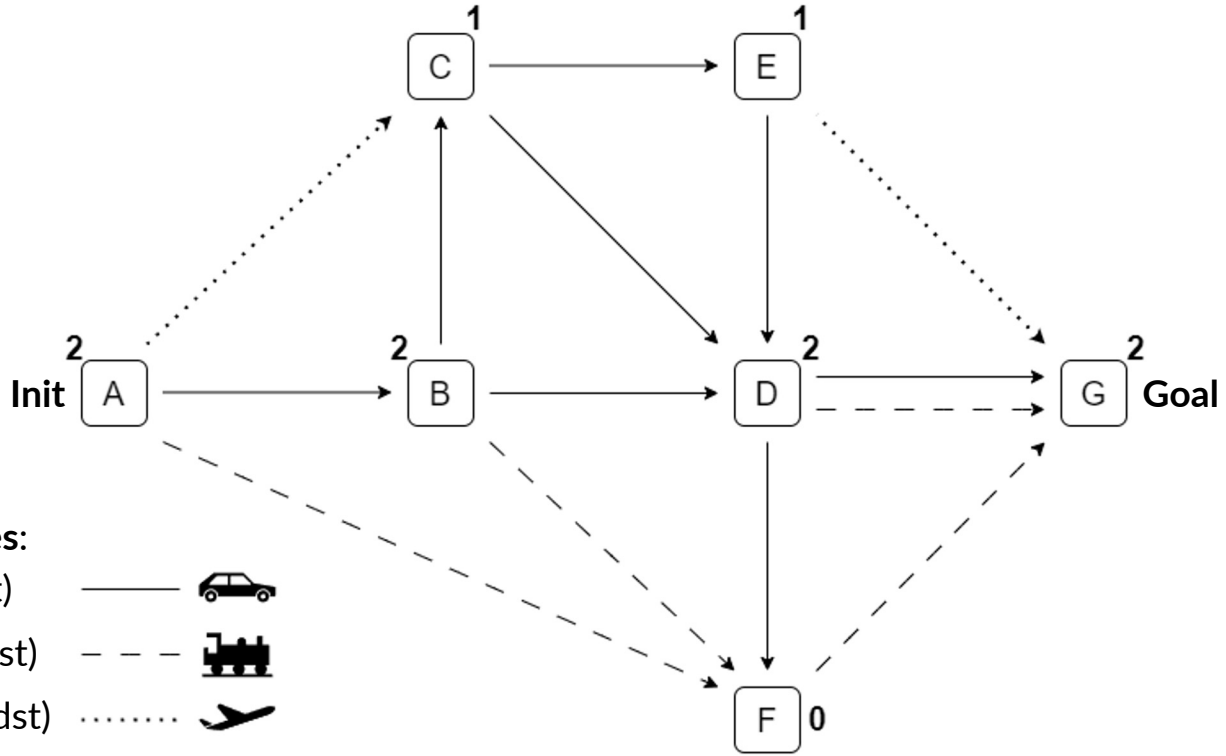
For a problem Π:

k-Resilient State: a state from which we can reach a goal state *even if k failures occur*:

1. A goal state is k-resilient.
2. A state *s* is 0-resilient if a plan from *s* to a goal state exists.
3. A state *s* is k-resilient if there exists an action *a* such that (1) its execution in *s* results in another k-resilient state, and (2) *s* is still (k-1)-resilient *without using a*.

K-Resilient plan solving <Π, K>**:** *a solution plan for Π where all traversed states are K-resilient.*

Illustrative Example (k-resilient state)



Action types:

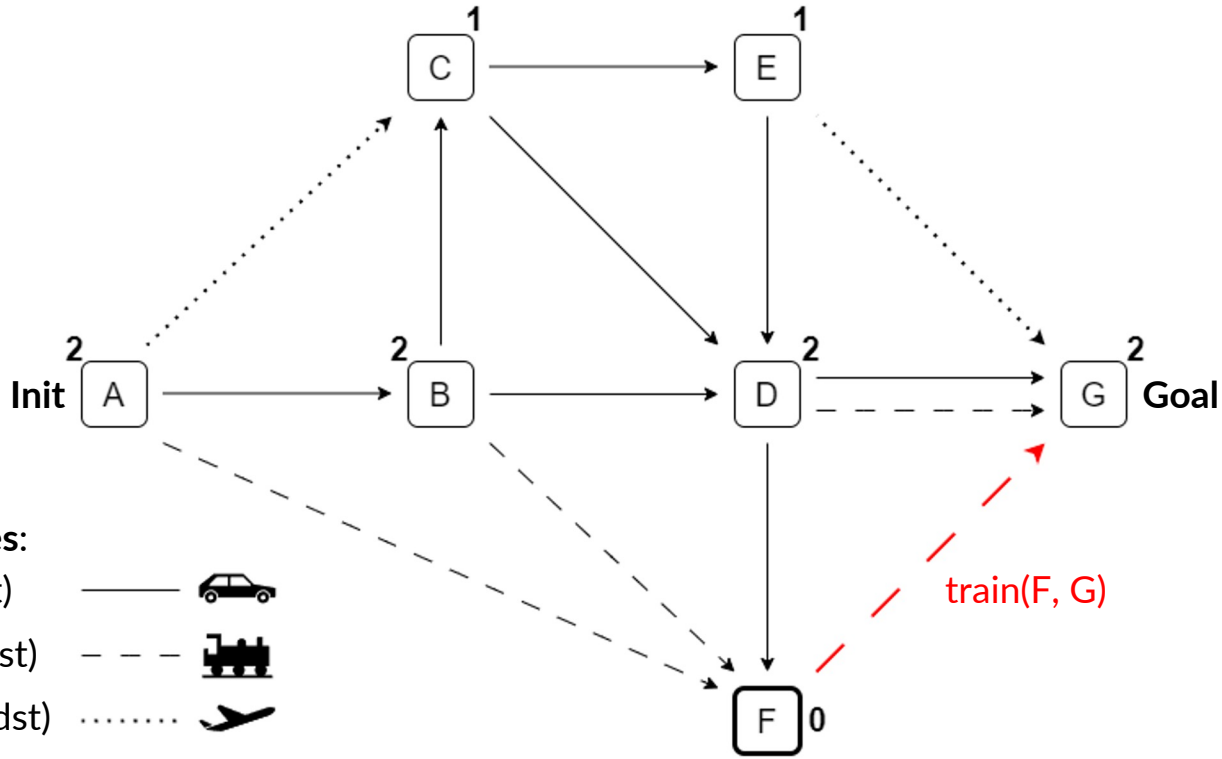
car(src, dst)

train(src, dst)

plane(src, dst)

States: A, B, ..., G marked by k-resilience values

Illustrative Example (k-resilient state)



Action types:

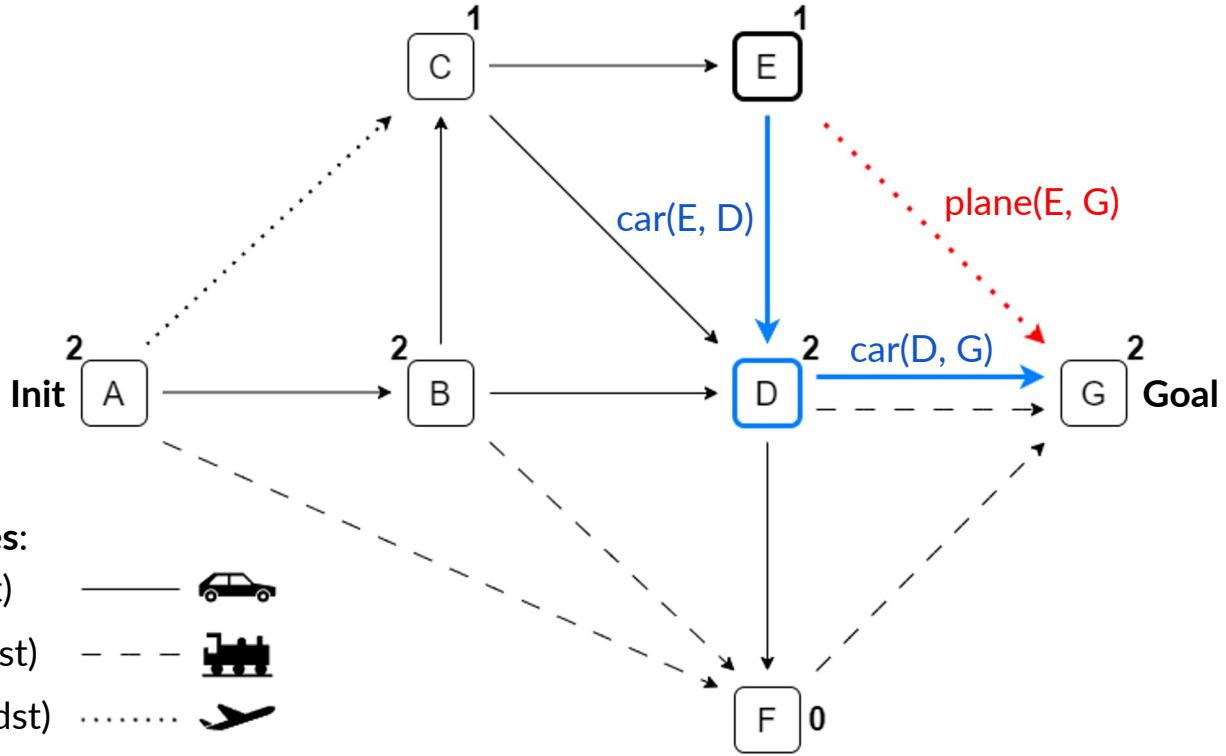
car(src, dst)

train(src, dst)

plane(src, dst)

States: A, B, ..., G marked by k-resilience values

Illustrative Example (k-resilient state)



Action types:

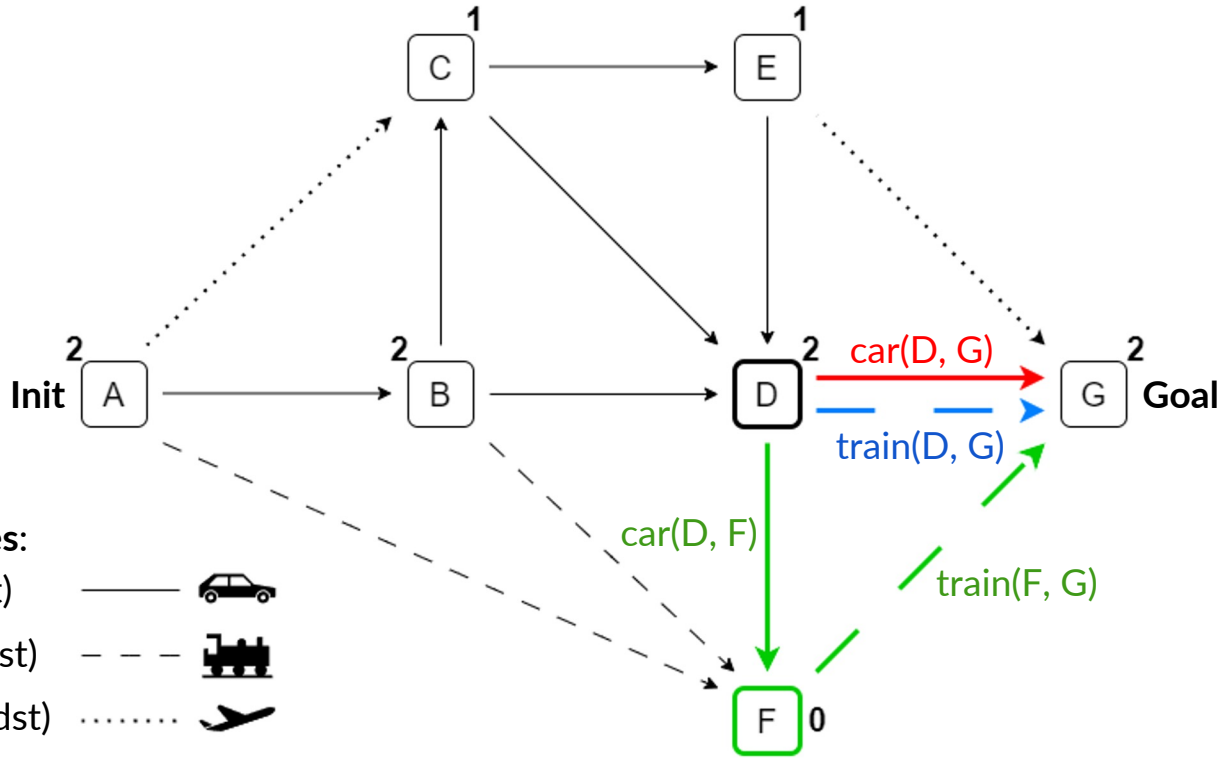
car(src, dst)

train(src, dst)

plane(src, dst)

States: A, B, ..., G marked by k-resilience values

Illustrative Example (k-resilient state)



Action types:

car(src, dst)

train(src, dst)

plane(src, dst)

States: A, B, ..., G marked by k-resilience values

ResPlan Algorithm and k-Resilience Checking

Method (high level): Find a plan π that solves Π and verify that every state traversed by π is *K*-resilient.

***k*-Resilience check <*s*, *k*, *V*>:** *is state **s** a **k**-resilient state in <F, A \ **V**, s_0, G>?*

Following the definition of ***k***-resilient state, it answers whether state ***s*** is ***k***-resilient without using the faulty actions ***V*** (forbidden after failure).

ResPlan and k-Resilience Checking

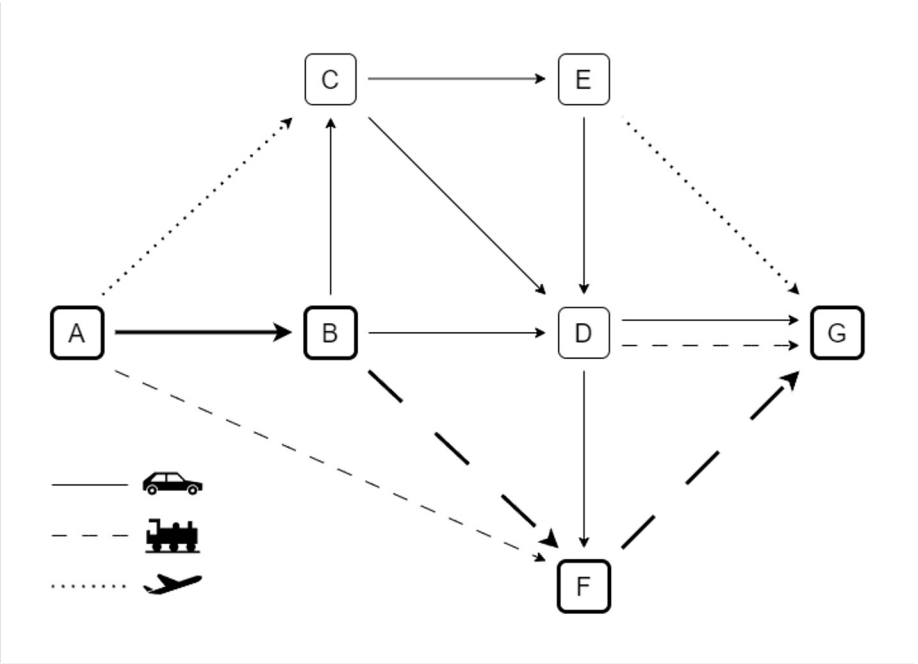
Plan under evaluation: car(A,B), train(B,F), train(F,G) **Is it 2-resilient?**

<G, 2, {}>:

<F, 2, {}>:

<B, 2, {}>:

<A, 2, {}>:



ResPlan and k-Resilience Checking

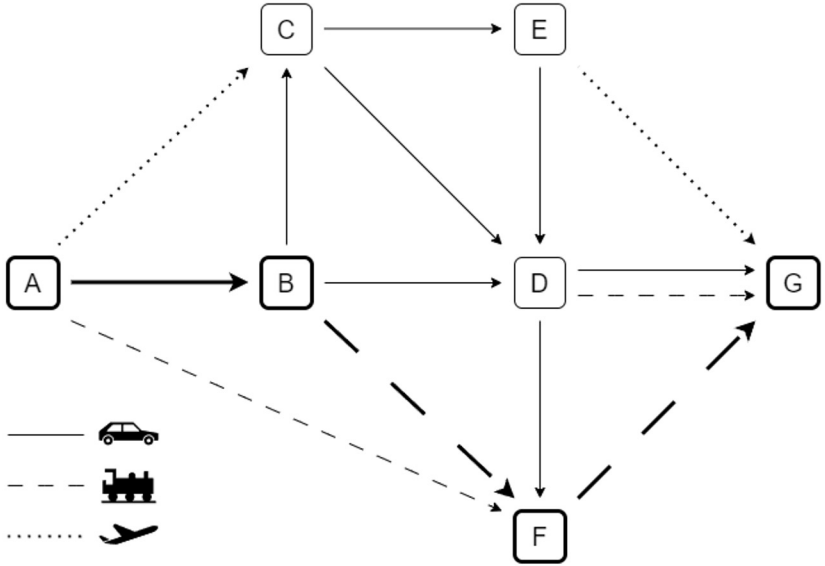
Plan under evaluation: car(A,B), train(B,F), train(F,G) **Is it 2-resilient?**

<G, 2, {}>: TRUE, G is a goal state

<F, 2, {}>:

<B, 2, {}>:

<A, 2, {}>:



ResPlan and k-Resilience Checking

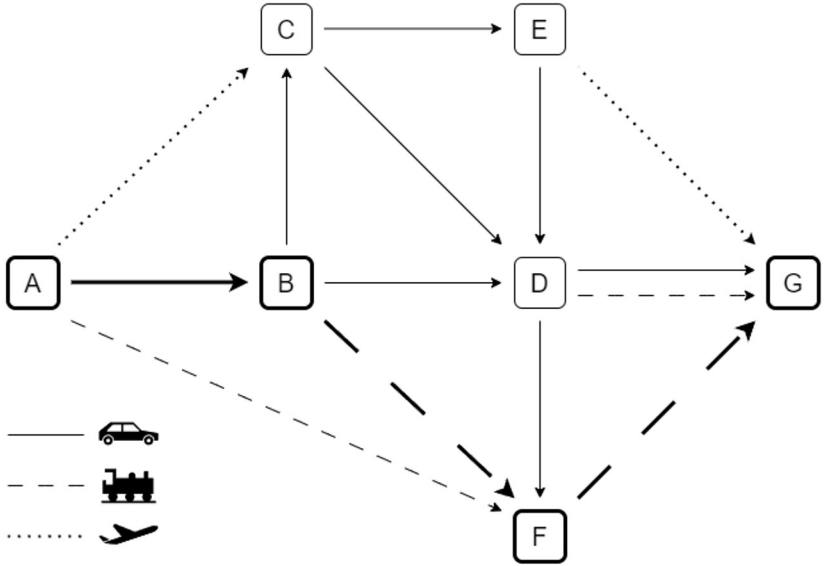
Plan under evaluation: car(A,B), train(B,F), train(F,G) **Is it 2-resilient?**

<G, 2, {}>: TRUE, G is a goal state

<F, 2, {}>:
<F[train(F,G)]=G, 2, {}>:
<F, 1, {train(F,G)}>:

<B, 2, {}>:

<A, 2, {}>:



ResPlan and k-Resilience Checking

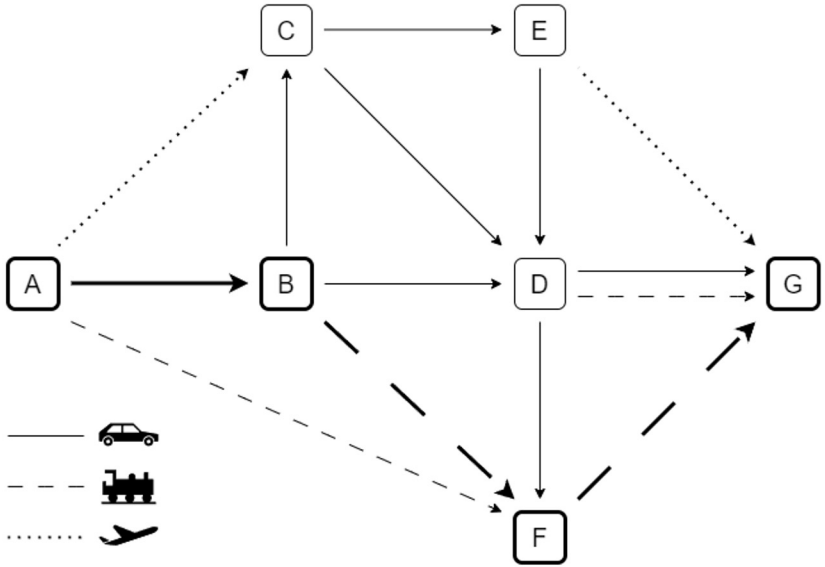
Plan under evaluation: car(A,B), train(B,F), train(F,G) **Is it 2-resilient? NO**

<G, 2, {}>: TRUE, G is a goal state

<F, 2, {}>:
<F[train(F,G)]=G, 2, {}>: TRUE
<F, 1, {train(F,G)}>: FALSE

<B, 2, {}>:

<A, 2, {}>:



ResPlan and k-Resilience Checking

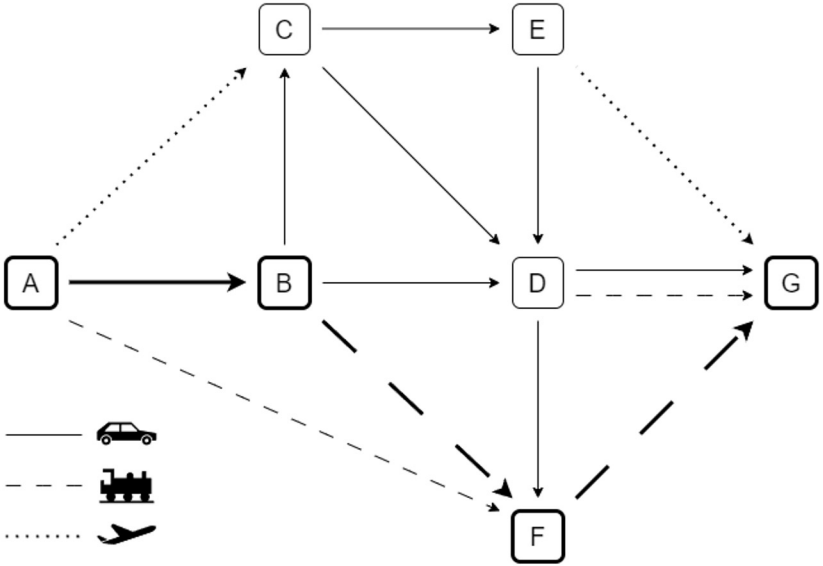
Partial plan under evaluation: car(A,B), ~~train(B,F), train(F,G)~~

<G, 2, {}>: TRUE, G is a goal state

<F, 2, {}>: FALSE
<F[train(F,G)]=G, 2, {}>: TRUE
<F, 1, {train(F,G)}>: FALSE

<B, 2, {}>:

<A, 2, {}>:



ResPlan and k-Resilience Checking

Partial plan under evaluation: *car(A,B), ~~train(B,F), train(F,G)~~*

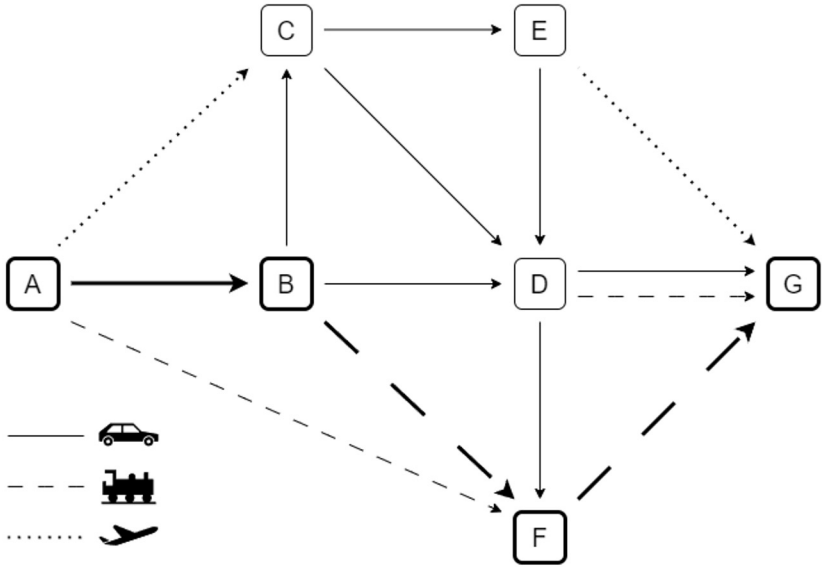
when (re)planning, avoid visiting states that we discovered ***not*** resilient

<G, 2, {}>: TRUE, G is a goal state

<F, 2, {}>: FALSE
<F[train(F,G)]=G, 2, {}>: TRUE
<F, 1, {train(F,G)}>: FALSE

<B, 2, {}>:

<A, 2, {}>:



ResPlan and k-Resilience Checking

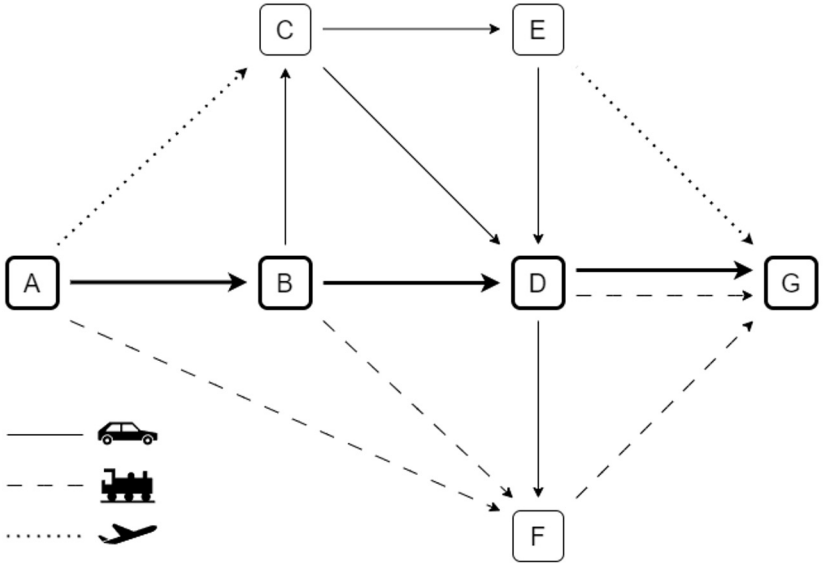
***Revised* plan under evaluation:** *car(A,B), car(B,D), car(D,G)* **Is it 2-resilient?**

<G, 2, {}>: TRUE, G is a goal state

<D, 2, {}>:

<B, 2, {}>:

<A, 2, {}>:



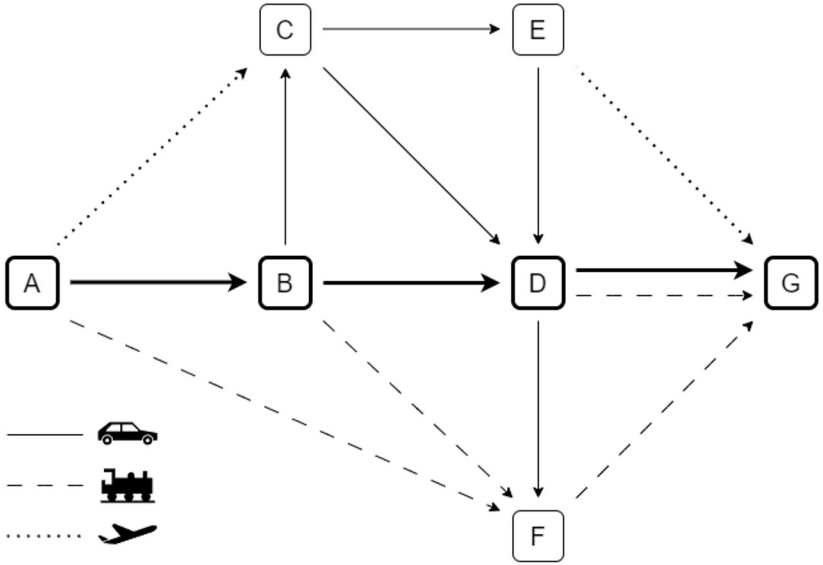
ResPlan and k-Resilience Checking

***Revised* plan under evaluation:** *car(A,B), car(B,D), car(D,G)* **Is it 2-resilient?**

<G, 2, {}>: TRUE, G is a goal state

<D, 2, {}>: TRUE
<D[car(D,G)]=G, 2, {}>: TRUE
<D, 1, {car(D,G)}>: TRUE
<D[train(D,G)]= G, 1, {car(D,G)>: TRUE
<D, 0, {train(D,G),car(D,G)>: TRUE

<B, 2, {}>:



<A, 2, {}>:

ResPlan and k-Resilience Checking

***Revised* plan under evaluation:** *car(A,B), car(B,D), car(D,G)* **Is it 2-resilient? YES**

<G, 2, {}>: TRUE, G is a goal state

<D, 2, {}>: TRUE
<D[car(D,G)]=G, 2, {}>: TRUE
<D, 1, {car(D,G)}>: TRUE
<D[train(D,G)]= G, 1, {car(D,G)>: TRUE
<D, 0, {train(D,G),car(D,G)>: TRUE

<B, 2, {}>: TRUE

<A, 2, {}>: TRUE