




# The Remote Agent Experiment

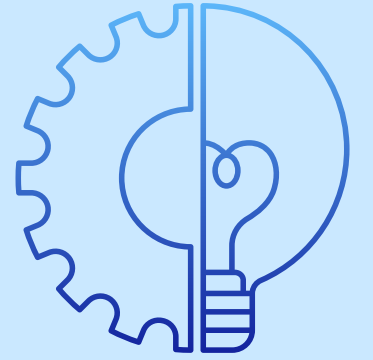
**Planejamento de viagens  
interplanetárias**

Gabriel Noal  
Lucca Nazari  
Mateus Amaral

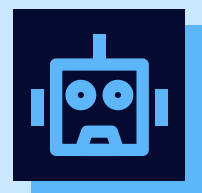




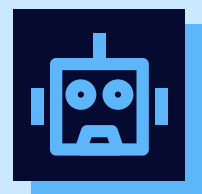
# Sobre o RAX...



- Primeira IA planejador/programador em execução em uma nave espacial - (17 Maio 1999)
- Execução através de um circuito fechado e inferência de estados.

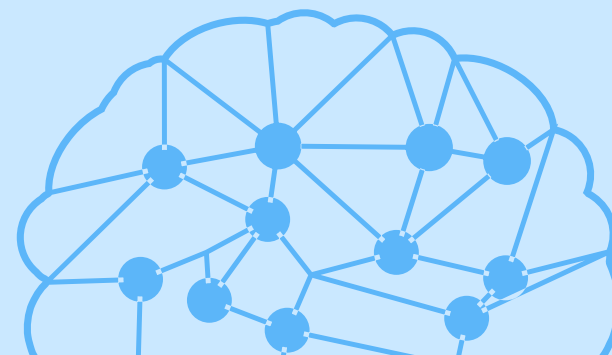
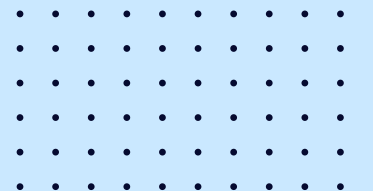


**Inferência de estado**

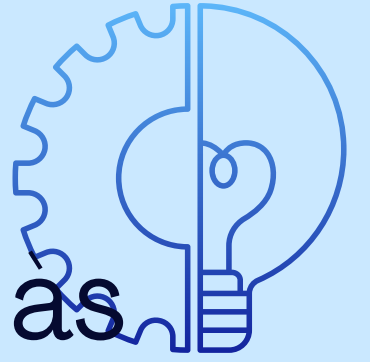


**Recuperação de falha**

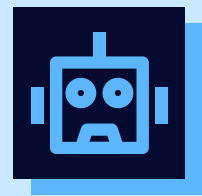
*O experimento conseguiu demonstrar a aplicabilidade do projeto e possibilitou sua ampliação para diferentes áreas*



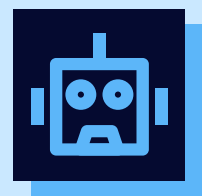
# RAX-PS



- Gera planos temporariamente flexíveis, permitindo ajustes as condições reais
- Planos são redes de restrições, construídas de forma incremental consultando o modelo da dinâmica da espaçonave (planejamento de intervalo baseado em restrições)

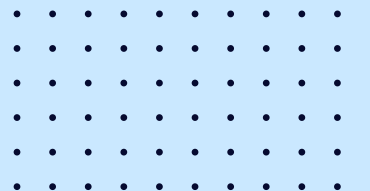
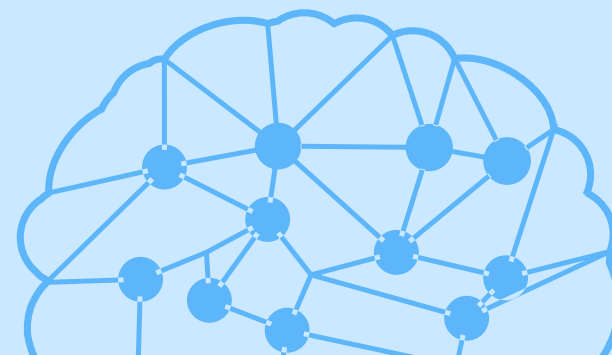


**Solicitação de planos**



**Geração de um plano**

*Utiliza tokens, linhas de tempo e variáveis de estado para caracterizar a realidade para que possa ser reproduzida.*





# Solicitação de planos



**O RAX-PS conduziu sua tomada de decisão com base nos objetivos principais da missão, dando origem a atividades de subobjetivo. No entanto, era necessário levar em conta alguns fatores:**

**01**

Segurança

**03**

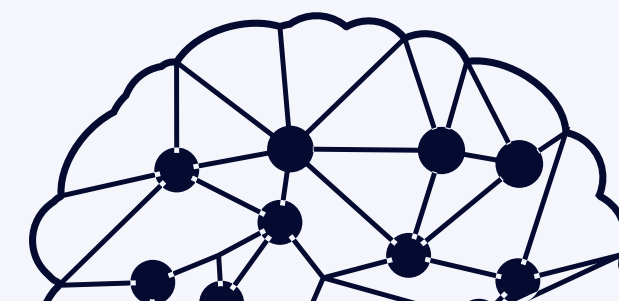
Controle de recursos

**02**

Duração de atividades

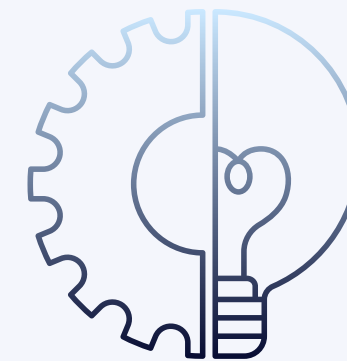
**04**

Interação de atividades





# Geração de planos



01

Inicializa o banco de dados do plano

02

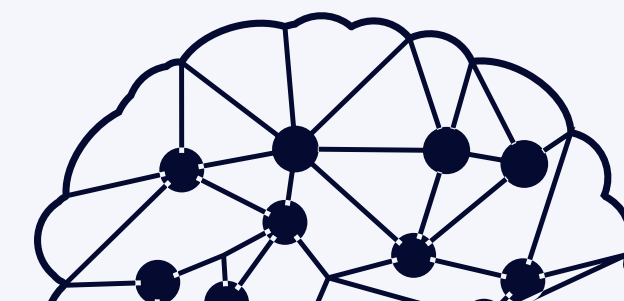
Heurística fornece orientação para o mecanismo de busca

03

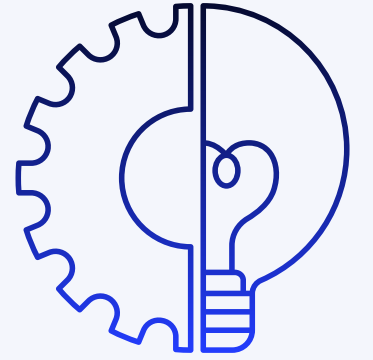
Mecanismo de busca modifica o banco de dados

04

Plano válido é enviado ao agente de execução



# Estrutura do planejamento



Uso de cronogramas para modelar e raciocinar sobre atividades simultâneas




Variáveis de estado representam atributos que mudam ao longo do tempo



O histórico de estados de uma variável de estado durante um período de tempo chama-se linha do tempo



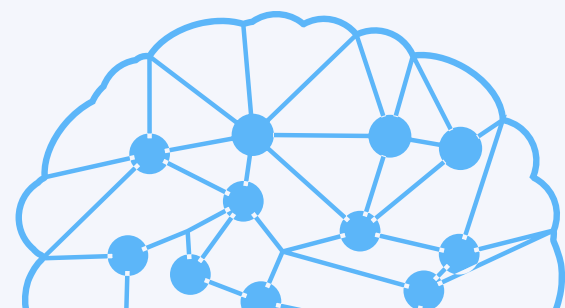
Um token descreve uma invocação do procedimento e seus parâmetros, as variáveis de estado nas quais pode ocorrer e os valores de tempo que definem seu intervalo




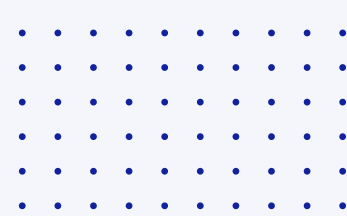

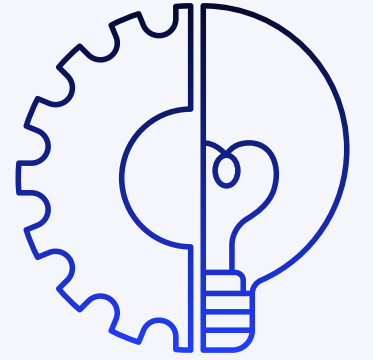
Token de restrição - associado a uma sequência de procedimentos, logo representa uma demanda de recursos (para o RAX, foi feita para modelar e acompanhar o uso de energia)




Token flutuante - um token que ainda não está em uma linha do tempo




# Restrições



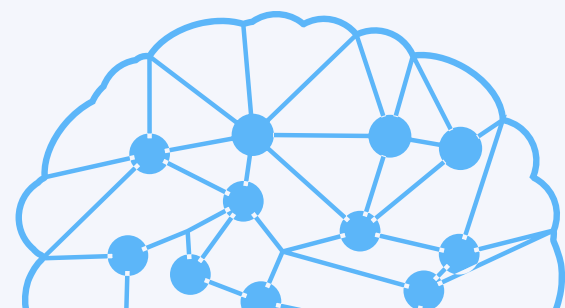
Uma chamada de procedimento pode funcionar somente após a conclusão de outro procedimento ou pode precisar ser executada em paralelo com um procedimento em um thread diferente



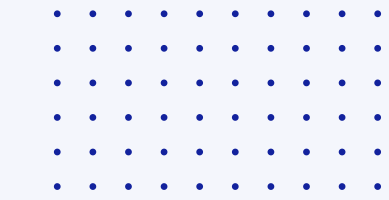
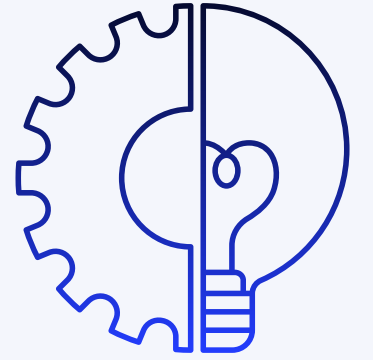
Compatibilidade - Determina quais procedimentos podem ser invocados e quais devem preceder, seguir, ser cotemporais, etc. É uma disjunção de restrições.



Restrições de subobjetivo - garantem que cada variável de estado esteja sempre executando um procedimento ou alternando instantaneamente entre invocações de procedimento



# Banco de Dados Plano



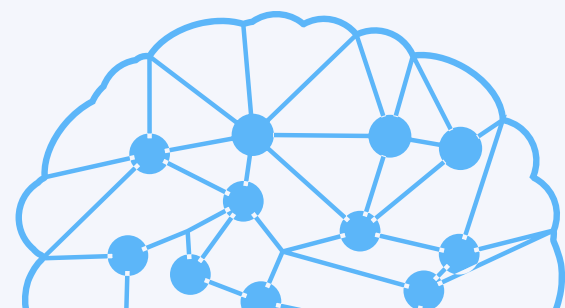
Plano candidato atual - conjunto de cronogramas contendo tokens relacionados entre si



Conjunto atual de decisões que necessitam ser tomadas. Uma decisão corresponde a uma falha em um plano candidato, um aspecto deste que pode impedir que seja um plano completo e válido

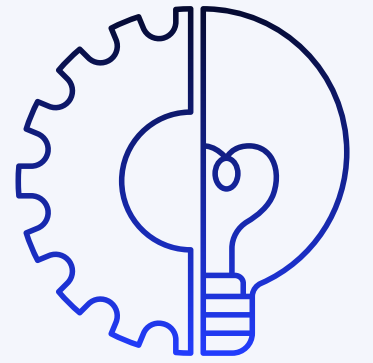


As restrições em um plano candidato dão origem a uma rede de restrições. Como resultado, qualquer plano candidato que tenha uma rede de restrição subjacente inconsistente não pode fazer parte de um plano válido.





# Falhas



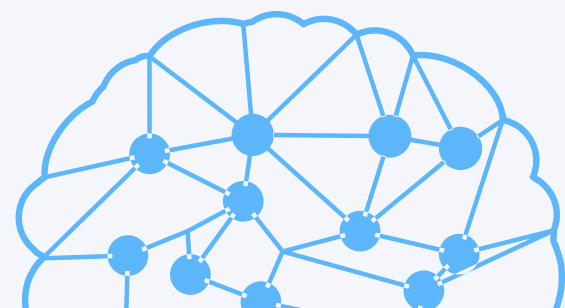
Uma decisão corresponde a uma falha em um plano candidato, um aspecto do candidato que pode impedir que seja um plano completo e válido

Existem quatro tipos de falhas: variáveis não instanciadas, tokens flutuantes, disjunções abertas de compatibilidades e subobjetivos de compatibilidade insatisfeitos

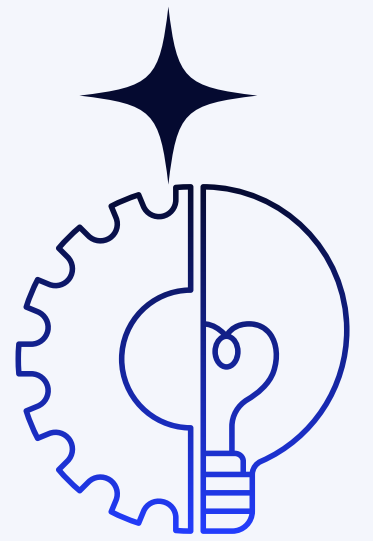
Cada falha no banco de dados do plano dá origem a escolhas de como essa falha pode ser resolvida

Resolver uma falha é uma etapa de raciocínio que mapeia o banco de dados fornecido para outro banco de dados

Não é necessário resolver todas as falhas para se ter um plano (flexibilidade dos planos), porém deseja-se que, na maioria dos casos, os subobjetivos de pelo menos uma das disjunções sejam satisfeitos



# Planos e comportamentos do sistema



## Plano - EXEC

Programa concorrente que deve ser interpretado e executado em um sistema dinâmico.

Deve conter variáveis que determinam como e quando os procedimentos devem ocorrer.



Disparada uma resposta de proteção contra falhas, caso os valores não correspondam aos esperados, após o EXEC.

## Sucesso do EXEC

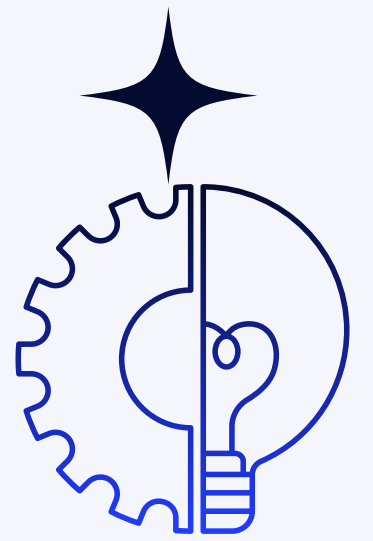
Para determinar se o Plano - EXEC foi bem sucedido, depende de dois fatores:

- 01 Capacidade de processamento
- 02 Tempo até a próxima tomada de decisão

Comparação entre valores reais do sistema com valores especificados, para determinar a ação.



# Planos e comportamentos do sistema



## Sucesso do EXEC

Caso o EXEC não tenha capacidade de processamento ou tempo suficiente para a tomada de decisão, a execução deve ser a mais simples possível.

■ Planos mais simples são as únicas evoluções possíveis do sistema



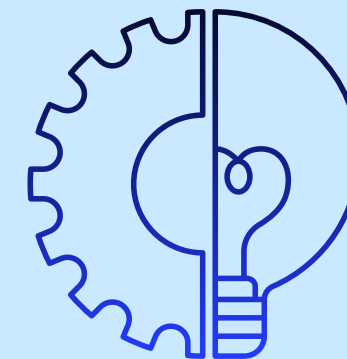
## Executabilidade

Depende de dois fatores:

- 01 Flexibilidade do plano candidato para cobrir as variações do sistema
- 02 Restrição do plano candidato para ser o EXEC e identificar se ele pode ser executado

Na prática, o planejamento sempre ocorre entre um horizonte finito de possibilidades

# Processo de Planejamento



A entrada para o processo de planejamento é um plano - candidato inicial.  
Incluindo:

**01** Token de inicialização para cada linha do tempo

**02** Conjunto de token flutuantes

**03** Conjunto de restrições para os tokens

Banco de dados do plano inicial (estado inicial + metas)

Objetivo:

Candidato inicial



Plano válido completo

O processo de planejamento é uma função recursiva que seleciona não deterministicamente uma resolução para uma falha no plano atual base de dados

Vantagens:

Sólido: qualquer plano resultante satisfaz a função dada

Completo: plano pode ser encontrado





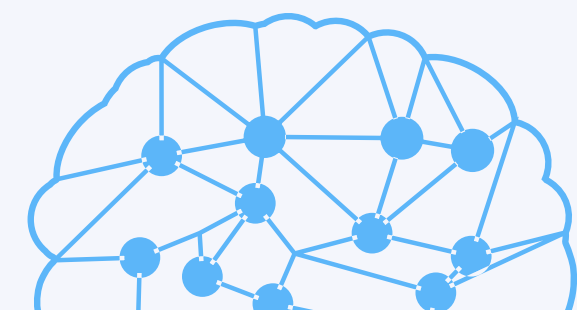
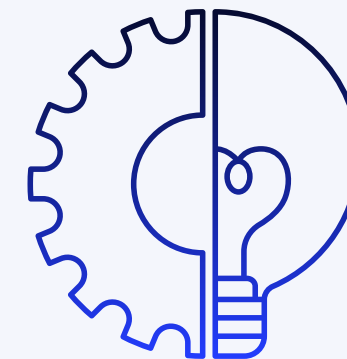
## Definição de um Oráculo

**Oráculo tem o objetivo de resolver cada falha que o processo de planejamento possa encontrar. Com o objetivo de encontrar um plano adequado**

- Conjunto de possíveis falhas é definido pelos tokens que aparecem no plano alvo.
- Para cada falha, o oráculo especifica como ela deve ser resolvida.

**Para comprovar que o oráculo resultará em um plano adequado, é necessário que:**

- 01** Todas as falhas necessárias para chegar a um plano final, apareçam.
- 02** Cada etapa fornece um candidato que pode ser estendido ao plano final.



## A Prática

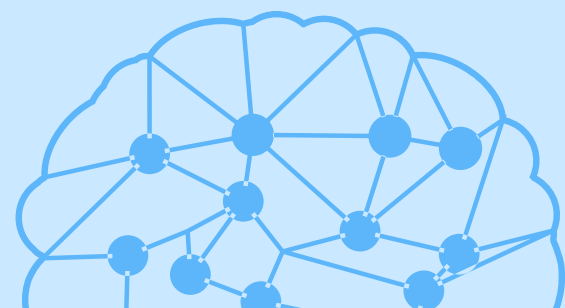
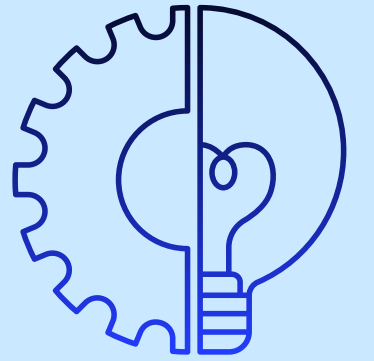
### Mecanismo de Planejamento RAX PS

- Planos RAX são flexíveis apenas na dimensão temporal
- Todas as variáveis devem ser vinculadas à um único valor

O controlador de busca do RAX PS permite programar um oráculo aproximado como uma lista de regras de controle de busca. A lista fornece uma priorização das falhas em um banco de dados e estratégias de classificação para as escolhas não determinísticas para cada seleção de falha.

### Gerenciamento da Agenda de Falhas

- O RAX fez uso de um controlador de pesquisa programável
- O controlador de busca 'ideal' é um "Oráculo"
- Porém, na prática, o controlador só pode tomar decisões de resolução de falhas, com base no plano desenvolvido até o momento da tomada de decisão.



# O Planejador em Voo

## Data

A data de lançamento foi em 17 de Maio de 1999.

## Objetivos

O experimento alcançou todos os objetivos de validação da tecnologia.

- No entanto, na manhã do dia 18, o Remote Agent parou de 'comandar' a espaçonave, enquanto ainda estava 'saudável'.
- Problema em uma condição de impasse de baixa probabilidade devido a uma seção crítica, que estava ausente no código EXEC.

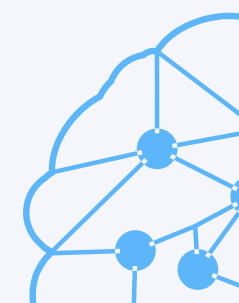
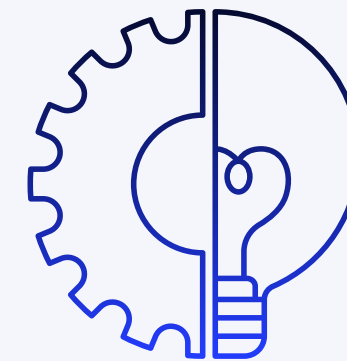
## Solução

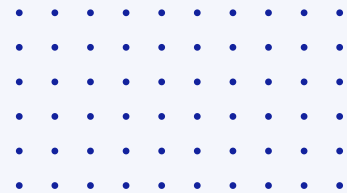
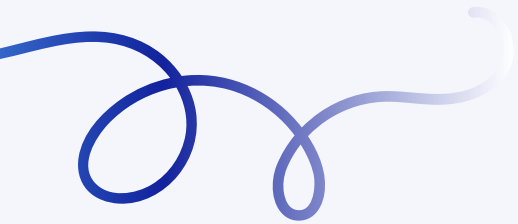
A equipe RAX, nas 10 horas seguintes, desenvolveu um cenário experimental novo para concluir a realização dos objetivos de validação do Remote Agent.

## Conclusão

A conclusão do experimento foi às 14:00 do dia 21 de maio de 2021. Atingindo 100% dos objetivos de validação propostos.

***"Uma falha de software potencialmente catastrófica acabou sendo uma vitrine inesperada de como a tecnologia de planejamento pode fortalecer e reduzir custos para futuras missões espaciais robóticas."***





# Q&A

