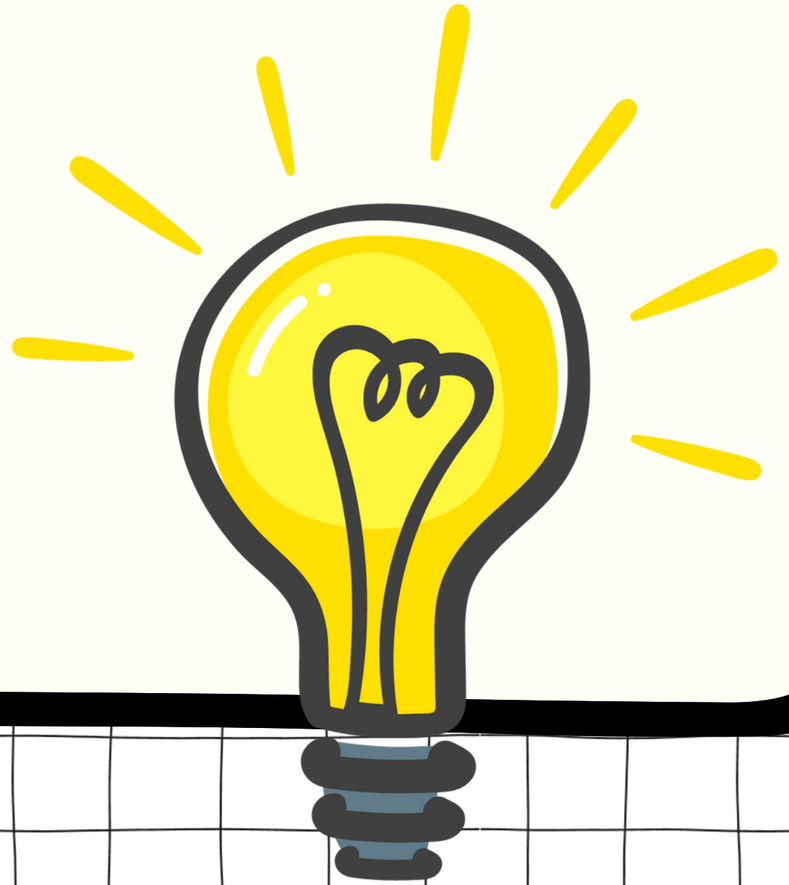
Factories

Apresentação do domínio do IPC 2023



Sobre

O domínio **Factories (simple)**, feito por *Dominik Schreiber e Malte Sonnichsen* para a IPC 2023, define um sistema de fábricas que dependem uma das outras para a fabricação de seus *recursos e produtos*.

O domínio faz parte da subcategoria HTN da competição, que utiliza a linguagem **HDDL**.

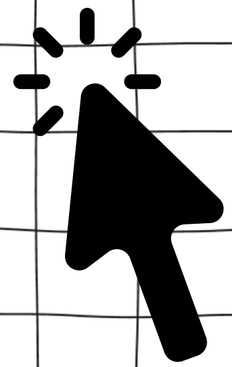
Métodos

01 Construção

02 Extração e fusão

03 Produção

04 Movimentação



01

Construção

01 Uma fábrica (*factory*) é responsável por produzir um certo tipo de **recurso**, dependendo de outras fábricas ou não.

02 Para a construção de uma nova fábrica, é necessário que haja os recursos requisitados, que a localização esteja livre e que a fábrica em questão não tenha sido construída previamente.

03 Caso os prérequisitos sejam atendidos, o planner, então, pegará os recursos necessários para a construção e construirá, de fato, a fábrica.

04 É comum que todo problema tenha pelo menos uma fábrica já construída ou que os recursos necessários para a construção de uma fábrica estejam disponíveis.

```
(:method m_construct_factory
        :parameters (?f - factory ?r - resource ?l - location)
        :task (construct_factory ?f ?l)
        :precondition (and
            (demands ?f ?r)
            (location-free ?l)
            (not (factory-constructed ?f))
        )
        :ordered-subtasks (and
            (get_resource ?r ?l)
            (construct ?f ?r ?l)
        )
    )
```

02

Extração

01 Cada fábrica é capaz de extrair e processar diferentes recursos para, então, **poderem serem utilizados por outras fábricas**.

02 Uma fábrica é capaz de extrair **um** recurso apenas.

03 Caso os prérequisitos sejam atendidos, o planner, então, será construída uma fábrica capaz de extrair este recurso, o produto então será produzido, depois, entregue para a próxima localização.

```
(:method m_get_resource
        :parameters (?r - resource ?f - factory ?fl ?l - location)
        :task (get_resource ?r ?l)
        :precondition (and
            (produces ?f ?r)
        )
        :ordered-subtasks (and
            (construct_factory ?f ?fl)
            (produce_resource ?r)
            (deliver_resource ?r ?l)
        )
    )
```

02

Fusão

01 Uma fábrica é capaz, também, de fundir - isto é, combinar - dois recursos diferentes para transformá-lo em um novo recurso.

02 Pode ser que um recurso qualquer não possa ser produzido diretamente em uma fábrica e requira a fusão de dois outros recursos para sua produção.

03 Caso os prérequisitos sejam atendidos, o planner, então, será obtido os dois recursos-base e, logo após, estes serão fundidos para formar o recurso final.

```
(:method m_get_resources_and_fuse
        :parameters (?r ?r1 ?r2 - resource ?l - location)
        :task (get_resource ?r ?l)
        :precondition (and
            (fuses ?r ?r1 ?r2)
        )
        :ordered-subtasks (and
            (get_resource ?r1 ?l)
            (get_resource ?r2 ?l)
            (fuse ?r ?r1 ?r2 ?l)
        )
    )
```

03

Produção

01 Uma fábrica tem o objetivo de produzir um recurso específico.

02 O processo de produção gera o produto final de uma fábrica.

03 Caso os prérequisitos sejam atendidos, o planner, então, será obtido o recurso-base para a produção - caso necessário, pois existem fábricas capazes de produzirem produtos sozinhas - e produzido o produto final.

```
(:method m_get_and_produce_resource
        :parameters (?r ?rd - resource ?f - factory ?l - location)
        :task (produce_resource ?r)
        :precondition (and
            (produces ?f ?r)
            (demands ?f ?rd)
            (factory-at ?f ?l)
        )
        :ordered-subtasks (and
            (get_resource ?rd ?l)
            (produce ?r ?rd ?f ?l)
        )
    )
```

04

Movimentação

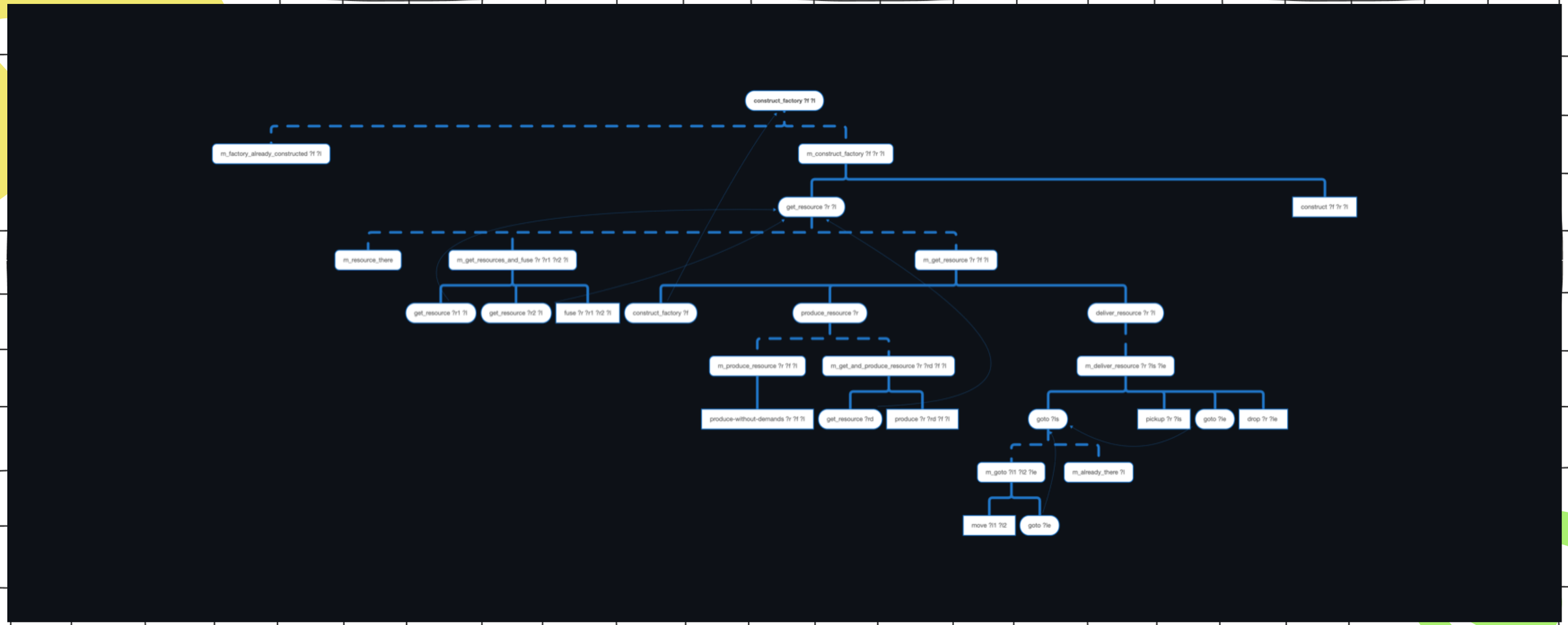
01 Fábricas utilizam caminhões para movimentar recursos para as localizações.

02 O método apresentado verifica se o caminhão está na localização e, se não, realoca-o para o lugar.

03 As condições para a movimentação é que o caminhão não já esteja no local e que seja possível travessar da localização inicial para a final.

```
(:method m_goto
        :parameters (?l1 ?l2 ?le - location)
        :task (goto ?le)
        :precondition (and
            (truck-at ?l1)
            (connected ?l1 ?l2)
        )
        :ordered-subtasks (and
            (move ?l1 ?l2)
            (goto ?le)
        )
    )
```

Tree


construct_factory ?f ?l
m_factory_already_constructed ?f ?l
m_construct_factory ?f ?r ?l
get_resource ?r ?l
construct ?f ?r ?l
m_resource_there
m_get_resources_and_fuse ?r ?r1 ?r2 ?l
m_get_resource ?r ?f ?l
get_resource ?r1 ?l
get_resource ?r2 ?l
fuse ?r ?r1 ?r2 ?l
construct_factory ?f
produce_resource ?r
deliver_resource ?r ?l
m_produce_resource ?r ?f ?l
m_get_and_produce_resource ?r ?rd ?f ?l
m_deliver_resource ?r ?ls ?le
produce-without-demands ?r ?f ?l
get_resource ?rd
produce ?r ?rd ?f ?l
goto ?ls
pickup ?r ?ls
goto ?le
drop ?r ?le
m_goto ?l1 ?l2 ?le
m_already_there ?l
move ?l1 ?l2
goto ?le

planner	total	Assembly	Barman BDI	Blocksworld HPDDL	Depots	Factories simple
PandaDealer-agile-lama	15.28808	0.89	0.78	0.77	0.90	0.67
lamda-to-agl-gbfs-ao	15.15338	1.00	0.84	0.84	0.90	0.42
PandaDealer-sat-1	14.89349	0.83	0.72	0.94	0.72	0.48
lamda-to-agl-gbfs-lmc	14.87160	0.89	0.80	0.71	0.87	0.42
ppro-to-sat-gbfs-add	13.62710	0.83	0.58	0.83	0.72	0.37
PandaDealer-sat-2	11.34366	0.20	0.50	0.30	0.80	0.35
toad-io-dfad	10.92964	1.00	0.71	0.73	0.83	0.25
toad-po-dfad	10.43419	1.00	0.71	0.73	0.77	0.25
toad-io-ff	10.23798	1.00	0.65	0.70	0.85	0.25
LiftedTreePath	7.12610	0.10	0.64	0.00	0.71	0.15
SIADEX	3.45194	0.00	0.61	0.00	0.73	0.00
aries-sat	3.19394	0.07	0.15	0.00	0.40	0.05
OptiPlan	0.72308	0.00	0.00	0.00	0.03	0.05