$Jobs(\{AddEngine1 \prec AddWheels1 \prec Inspect1\},$
$\{AddEngine2 \prec AddWheels2 \prec Inspect2\})$

$Resources(EngineHoists(1), WheelStations(1), Inspectors(2), LugNuts(500))$

$Action(AddEngine1,$ DURATION:30,
USE:$EngineHoists(1))$
$Action(AddEngine2,$ DURATION:60,
USE:$EngineHoists(1))$
$Action(AddWheels1,$ DURATION:30,
CONSUME:$LugNuts(20)$, USE:$WheelStations(1))$
$Action(AddWheels2,$ DURATION:15,
CONSUME:$LugNuts(20)$, USE:$WheelStations(1))$
$Action(Inspect_i,$ DURATION:10,
USE:$Inspectors(1))$

Figure 11.13 A job-shop scheduling problem for assembling two cars, with resource constraints. The notation $A \prec B$ means that action A must precede action B.

The approach we take is "plan first, schedule later": divide the overall problem into a *planning* phase in which actions are selected, with some ordering constraints, to meet the goals of the problem, and a later *scheduling* phase, in which temporal information is added to the plan to ensure that it meets resource and deadline constraints. This approach is common in real-world manufacturing and logistical settings, where the planning phase is sometimes automated, and sometimes performed by human experts.

11.6.1 Representing temporal and resource constraints

Job-shop scheduling problem
Job
Duration
Consumable
Reusable

A typical **job-shop scheduling problem** (see Section 6.1.2), consists of a set of **jobs**, each of which has a collection of **actions** with ordering constraints among them. Each action has a **duration** and a set of resource constraints required by the action. A constraint specifies a *type* of resource (e.g., bolts, wrenches, or pilots), the number of that resource required, and whether that resource is **consumable** (e.g., the bolts are no longer available for use) or **reusable** (e.g., a pilot is occupied during a flight but is available again when the flight is over). Actions can also produce resources (e.g., manufacturing and resupply actions).

A solution to a job-shop scheduling problem specifies the start times for each action and must satisfy all the temporal ordering constraints and resource constraints. As with search and planning problems, solutions can be evaluated according to a cost function; this can be quite complicated, with nonlinear resource costs, time-dependent delay costs, and so on. For simplicity, we assume that the cost function is just the total duration of the plan, which is called the **makespan**.

Makespan

Figure 11.13 shows a simple example: a problem involving the assembly of two cars. The problem consists of two jobs, each of the form $[AddEngine, AddWheels, Inspect]$. Then the *Resources* statement declares that there are four types of resources, and gives the number of each type available at the start: 1 engine hoist, 1 wheel station, 2 inspectors, and 500 lug nuts. The action schemas give the duration and resource needs of each action. The lug nuts