From STRIPS to ADL and PDDL

STRIPS was first developed as an automated planning system (Fikes and Nilsson, 1971). The name was later referred to an action language for expressing and formulizing a planning problem. Probably, it's the most import base language for the later more advanced variants of action languages.

A STRIPS instance contains at least three components: initial state, goal state and action. Action is composed of precondition, post-condition (or effect) and the action name itself. STRIPS is a closed world, anything not occurring in the condition is assumed false. It only allows positive literals and conjunctions, but not negative literals or disjunctions, Instead, it may employ two data structures. One of positive literals and the other of negative literals. In STRIPS, it's difficult to model a group of objects with the same property or type. For example, if we need to model 100 people then we'll need to have 100 expressions for them.

As an improvement of STRIPS over the mentioned limitations, ADL (Pednault, 1986) allows negative literals and disjunctions as well. It's open world, meaning anything not occurring in the preconditions is considered as unknown, which is more towards our real-world perceptions. In addition to that, ADL supports types and quantified variables which makes it easy to model many objects of the same type. Conditional effect is also allowed in ADL while STRIPS can only do conjunctions of effects. It makes ADL much neat and concise to express conditional effect.

After years' of development and advancement, PDDL (Ghallab and etcs, 1998) was introduced as a computer-parsable, standardized syntax for representing STRIPS, ADL and other languages. The biggest change over the earlier various action languages is separating a planning problem into domain description and problem description. The common elements in every specific problem will be contained in the domain description while elements in the specific problem will be contained in the problem description. Therefore, several problem descriptions can be connected to the same domain description. It's much like the relationship of class and instance in the object-oriented programming.

There have been quite several new releases since the first version of PDDL in 1998. And many successors/variants/extensions are started to emerge since year 2002 as well. They are only possible because of the extensive study and research of action language since almost half a century ago.