Pronto offers core OWL reasoning services for knowledge bases containing uncertain knowledge; that is, it processes statements like "Bird is a subclass-of Flying Object with probability greater than 90%" or "Tweety is-a Flying Object with probability less than 5%". The use cases for Pronto include ontology and data alignment, as well as reasoning about uncertain domain knowledge generally, for example, risk factors associated with breast cancer.

First, I should say that if you are interested in a rigorous description of the approach, read the paper by Thomas Lukasiewicz "Probabilistic Description Logics for the Semantic Web" [1]. Pronto is to a large extent an implementation of the Lukasiewicz approach—the rest is optimization and the support of explanations.

In a nutshell, the features of Pronto (in addition to the features of Pellet[2]) are the following:

1. Expressing *generic probabilistic knowledge*. "Generic" means that the knowledge doesn't apply to any specific individual but rather to a fresh, randomly chosen one. Generic probabilistic knowledge is represented in the form of generic conditional constraint (GCC). A GCC is an expression of the form (D|C)[l,u], where C and D are DL concepts and [l,u] is a closed subinterval of [0,1]. Without getting deeply into the semantics, the meaning of a GCC is roughly *for a randomly chosen instance of C, the probability of being an instance of D is within [l,u].* The above statement about birds would be written as (FlyingObject|Bird)[0.9;1.0].

2. Expressing *concrete probabilistic knowledge*. Here the knowledge applies to a specific individual. Concrete probabilistic knowledge is represented in the form of a:X, where "a" is an individual and "X" is a GCC restricted to the form (D|owl:Thing)[l,u]. We can express "Tweety is-a Flying Object with probability less than 5%" as Tweety:(FlyingObject|owl:Thing)[0.0;0.05].

3. Probabilistic reasoning, that is, generic and concrete entailments. A generic entailment is, given a probabilistic KB and a pair of concepts, compute the tightest interval (D|C)[l,u]. A concrete entailment is, given a probabilistic KB, an individual "a", and a concept "D", compute the tightest interval (D|owl:Thing)[l,u] for "a". So we can ask Pronto to infer the probability of a statement like Tweety being a flying object based on other statements rather than asserting the conditional constraint.

4. Probabilistic explanations. Pronto is capable of computing all minimally sufficient (w.r.t. inclusion) subsets of conditional constraints for a particular entailment, both generic and concrete.

Perhaps the single most important point about Pronto reasoning is that all inferences are done in a totally "logical" way, i.e. using a well-defined entailment relation and without any explicit or implicit translation of KB (or some parts of KB) to Bayesian graphs. This is the major difference between Pronto and other approaches, e.g. P-CLASSIC [3] or "Probabilistic Extension to OWL" [4].

Finally, I should mention *overriding* as a feature of Pronto that we particularly like. Pronto allows certain conflicts between different pieces of probabilistic knowledge, more precisely, between different conditional constraints. The famous example is that of flying birds and non-flying penguins. The problem here is related to non-monotonicity: A bird is a flying object with high probability and all penguins are birds but a penguin has a low probability of flying.

The way Pronto resolves these conflicts is by allowing more specific constraints to *override* more generic ones. So if Pronto knows that Tweety is a Penguin and Penguin is a subclass-of Bird, it will override the constraint (FlyingObject|Bird)[0.9;1.0] by (FlyingObject|Penguin)[0.0;0.05] and correctly entail

Tweety:(FlyingObject|owl:Thing)[0.0;0.05]. This is the idea borrowed from *reference class reasoning* and supported by Lehmann's lexicographic entailment employed in Pronto (see the Lukasiewicz paper for technical details). The decision whether a constraint is more specific/generic than some other one is made through the classical DL reasoning.

Resources:
[1] Lukasiewicz, T.: Probabilistic description logics for the semantic web. Technical Report Nr. 1843-06-05, Institut fur Informationssysteme, Technische Universitat Wien (2007), available at: http://www.kr.tuwien.ac.at/research/reports/rr0605.pdf
[2] Features of Pellet, available at: http://pellet.owldl.com/features
[3] Koller, D., Levy, A., Pfeffer, A.: P-CLASSIC: a tractable probabilistic description logic. In: AAAI-1997, AAAI Press / MIT Press (1997) 390–397
[4] Ding, Z. and Peng, Y.: A Probabilistic Extension to Ontology Language OWL, available at: http://ebiquity.umbc.edu/paper/html/id/92/