Question: What benefits have you gained from these teachings?

All the engineers interviewed have taken the courses of the formal track. T. has not taken MFDLS. None of them apply formal methods in their profesionnal life, even though . T. is not so far, since this engineer is developing a SAT solver.

E.
For modules related to formal methods, it is true that their teaching does not apply to my current professional life. I think one of the reasons is that I did not have yet the opportunity to work on projects that are sufficiently critical, to the point of requiring these kinds of methods to ensure that they work properly. Indeed, since formal methods are quite expensive (especially in terms of time) to set up, they rarely fit into the priorities of companies developing services.

However, I think that these modules have brought me, on the one hand, a better understanding of the programming languages and correctness mechanisms integrated into these languages (typing systems, monitoring the life cycle of variables, etc.) and on the other hand, the knowledge of tools that can be used to prove programs (use of a proof language then implementation in a classical language, static analysis of existing programs, etc.).

Like most of the courses at ENSIIE, I appreciated the combination of purely theoretical knowledge and their rapid implementation in the form of projects, which makes it possible to better perceive how these methods can be used in a professional setting. I also appreciated the bridge that these courses place between computer programs and mathematics, in particular proof.

A.
Formal methods gave me rigor in software design. I think they helped me in my support function: being able to easily recognize pieces of faulty code, I can debug faster.
In software development, I also benefit from a certain ease to write unit and functional tests.

V.
It is true that formal methods are not the core of my daily work. As far as the notions seen in MFDLS are concerned, I do not think of using them directly or indirectly in my daily work.
As far as the notions of PROG1 / PROG2 are concerned, I sometimes use some of them, more or less regularly.

For example, recently, as part of my job, I added a new feature to a relatively poorly designed, poorly documented program that nobody could really tell me about. So I developed a sort of control flow graph (a little simplified anyway) to visualize the different states and understand the general operation of the program. This allowed me to develop the features requested by inserting my code in the appropriate places, while limiting the introduction of bugs in the initial behavior of the program.

I also do not pay attention to the notions I have learned in writing tests for my programs and the functions that compose them. I use, for example, data flow graphs to find suitable values to provide input of my functions to test a specific behavior. I can also use the notions of variants or invariants, but it is quite rare in Java development.

In general, I think that all the notions learned about analysis of a program, its source code, and its operation, allow us to better understand what we are developing, to better understand what is happening when we write this or that instruction in our code.

S.

Formal methods bring and require rigor and a rigor in reasoning.
Thus this rigor applied to the various projects that can be encountered in business becomes a real advantage. Proceed step by step, as for a proof, makes it possible to forget nothing, and to protect oneself upstream of possible future risks; risks that if they are realized often have a direct cost / impact on the project: late delivery, product not confirmed if necessary ...

T.

I did not follow the MFDLS elective course so my answer would be more focused on Prog 1 and more generally on logic courses.
These courses have brought me a lot from a technical point of view. As a developer using Java and C ++ technologies, it is difficult to become aware of reasoned programming that is much safer and therefore reliable. Despite my lack of practice in this area, the knowledge of these notions Lambda Calculus, software verification, methods of proof, has allowed me to repeatedly "shine" in professional interviews. Fascinated by aviation, I have postulated a lot in the aeronautics area, which has a strong demand for this safe programming and just vaguely addressing these topics has allowed me to stand out and have good feedback on my interview.