Is this "the first" programming languages paper? Either way, Landin clarifies problems in language design [1]. He isolates two main components of a language definition:

- *Core forms*: the basic primitives available to language users.
- *Derived forms*: ways of expressing new forms or structures in terms of core forms.

Landin observes that the core forms really define a language, while the derived forms (or rather, the general machinery for building derived forms) are essentially the same across languages. Thus he advocates for families of languages that share derived forms but offer different core forms suited to a specific user-base.

This is essential language design.

Strengths

- The discussion of denotations and sub-languages with algebraic laws was interesting and very relevant today. Theorem provers are one example; in that domain, the algebraic laws happen to enable very precise reasoning about program correctness. I especially liked the comments at the end, in particular Strachey's:
  - Instead of building little languages, we could use a technique for reasoning about imperative commands.
  - The imperative definition of matrix multiplication takes far too much detail—this makes refactoring and optimization difficult for computer and programmer alike.
- Thank you, Peter J. Landin for wading through so many languages and reference manuals, then stepping back and identifying the common structure.

Weaknesses

- The ALGOL 60 report used BNF notation, and I wish Landin had done the same for the figure in Section 5.
- Section 4 was unclear to me (perhaps, my own weakness) and I would have liked more clarification.
- The discussion of "relevant whitespace" in the text and during the Q/A notes is silly, and it's a shame we have the same silly discussions today.

References

[1] Peter J. Landin. The next 700 programming languages. In *CACM*, 1966.