Relazione sull'Utilizzo di Strumenti di Intelligenza Artificiale Generativa

1. Strumenti/LLM Utilizzati

Per lo sviluppo di questo progetto è stato utilizzato principalmente **Google Gemini** (modello Advanced). Lo strumento è stato impiegato come "Pair Programmer" interattivo e come supporto per la documentazione tecnica.

2. Casi Specifici di Impiego

L'integrazione dell'AI è avvenuta in fasi distinte del ciclo di vita del software, supportando una logica progettuale pre-esistente:

- **Raffinamento delle Specifiche:** Supporto nella revisione del documento di specifica originale ("Csar") per adattarlo ai requisiti "Lite" suggeriti dal docente, garantendo coerenza tra le regole grammaticali.
- **Setup dell'Ambiente:** Assistenza nella configurazione dell'ambiente Python, specificamente per la gestione delle dipendenze tra `Lark` (parser) e `llvmlite` (binding LLVM), risolvendo problematiche di compatibilità iniziale.
- **Generazione di Codice "Boilerplate":** Creazione rapida delle classi per i Nodi dell'AST (`ast_nodes.py`) e della struttura base dei Visitor Pattern, riducendo il tempo dedicato alla scrittura di codice ripetitivo.
- **Generazione di Test Case:** Creazione di programmi di esempio in linguaggio Csar (es. calcolo fattoriale, fibonacci) per testare le funzionalità del compilatore durante lo sviluppo.
- **Debugging e Analisi Errori:** Analisi dei traceback di Python e, soprattutto, interpretazione degli errori criptici generati da LLVM IR in fase di `verify()`. L'AI ha aiutato a individuare disallineamenti tra i tipi (es. `i32` vs `i1`) nel backend.

3. Vantaggi e Svantaggi Riscontrati

Vantaggi:

- **Velocità di Prototipazione**: La traduzione dalla grammatica EBNF alle regole regex di `Lark` è stata immediata, permettendo di avere un parser funzionante in tempi ridottissimi.
- **Focus sulla Logica:** Delegando all'AI la scrittura della sintassi Python più verbosa (es. definizioni di dataclass), è stato possibile concentrarsi sulla logica semantica e sulla gestione della Symbol Table.
- **Apprendimento di LLVM:** L'AI ha agito come un tutor per spiegare il funzionamento delle istruzioni IR (come `alloca`, `store`, `phi nodes`), colmando il divario tra la teoria studiata e l'implementazione pratica.

Svantaggi:

- **Allucinazioni sul Codice:** In alcuni casi, l'AI ha suggerito metodi di `llvmlite` deprecati o inesistenti, richiedendo una verifica manuale sulla documentazione ufficiale.
- **Complessità del Debugging Logico:** Quando l'AI generava codice sintatticamente corretto ma logicamente errato (es. gestione sbagliata dell'annidamento degli scope), il debugging è risultato più complesso, richiedendo una profonda conoscenza teorica per individuare l'errore "nascosto".

4. Lezioni Imparate e Commenti Personali

L'adozione di strumenti AI ha dimostrato che questi modelli sono eccellenti acceleratori di produttività, ma l'esperienza ha anche sollevato riflessioni critiche riguardanti il rischio di **deskilling professionale**.

Abbiamo percepito chiaramente come l'uso eccessivo di questi assistenti possa portare a una sorta di "impigrimento" intellettuale. Il pericolo è quello di abituarsi a ricevere soluzioni pronte, saltando la fase di ragionamento e "struggle" cognitivo che è fondamentale per l'apprendimento profondo e la memoria a lungo termine. Se ci si limita a fare copia-incolla, si rischia di diventare operatori passivi che non sanno più scrivere codice da zero o comprendere le logiche di basso livello.

Tuttavia, nel contesto di questo progetto, abbiamo mitigato questo rischio anteponendo lo studio teorico all'uso del tool. La logica del compilatore (fasi di analisi, costruzione AST, Type Checking) doveva essere ben chiara *prima* di interrogare il modello. Senza una solida base di teoria dei compilatori, sarebbe stato impossibile:

1. **Formulare i prompt corretti** per guidare la generazione del codice (sapere *cosa* chiedere).
2. **Validare l'output prodotto**, distinguendo codice funzionante da allucinazioni o implementazioni inefficienti.
3. **Mantenere la ownership del progetto**, usando l'AI come un "consulente" e non come un sostituto delle mie competenze.

In conclusione, l'AI ha trasformato il progetto in un esercizio di **architettura software e revisione critica**, permettendomi di realizzare un compilatore completo nei tempi previsti, ma ha anche rafforzato la mia convinzione che la competenza umana resti insostituibile per la validazione e la comprensione profonda dei sistemi complessi.