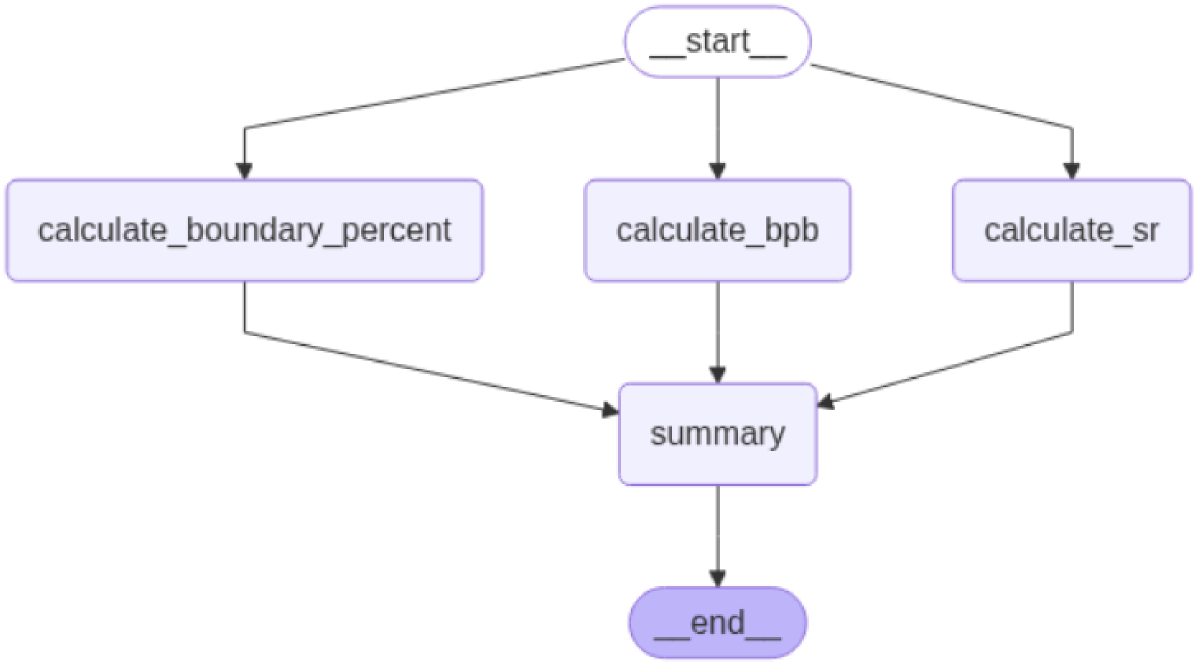
Parallel Workflows in LangGraph

Non-LLM WF

- Input : Batsman data
- Output : SR, Runs in Boundary %, balls per boundary
  - This can be calculated parallely
- Generate a **summary**



```
from langgraph.graph import StateGraph, START, END
from typing import TypedDict
```

State:

```
classBatsmanState(TypedDict):

    runs: int
    balls: int
    fours: int
    sixes: int

    sr: float
    bpb: float
    boundary_percent: float
    summary: str
```

Nodes:

```
# Calculate SR
def calculate_sr(state: BatsmanState):

    sr = (state['runs']/state['balls'])*100

    return {'sr': sr}
```

```
# Calculate Balls per Boundary
def calculate_bpb(state: BatsmanState):

    bpb = state['balls']/(state['fours'] + state['sixes'])

    return {'bpb': bpb}
```

```
# Calculate bondary %

def calculate_boundary_percent(state: BatsmanState):

    boundary_percent = (((state['fours'] * 4) + (state['sixes'] * 6))/state['runs'])
```

```
*100

    return {'boundary_percent': boundary_percent}
```

```
# Sumaary

def summary(state: BatsmanState):

    summary = f"""
Strike Rate - {state['sr']} \n
Balls per boundary - {state['bpb']} \n
Boundary percent - {state['boundary_percent']}
"""

    return {'summary': summary}
```

💡 **!!VIMP: While working in Parallel wfs, DO NOT RETURN STATE. RETURN PARTIAL STATE.**

> We'll return a **DICTIONARY**

- The function expects a dictionary. But we can't return state.
    - So, we're returning the parameter we changed in dictionary.

❗ Why this breaks?

- Both nodes modify the **same key**
- LangGraph does **not know which one wins**
- Result → ❌ conflict / overwrite / error

The correct mental model (very important)

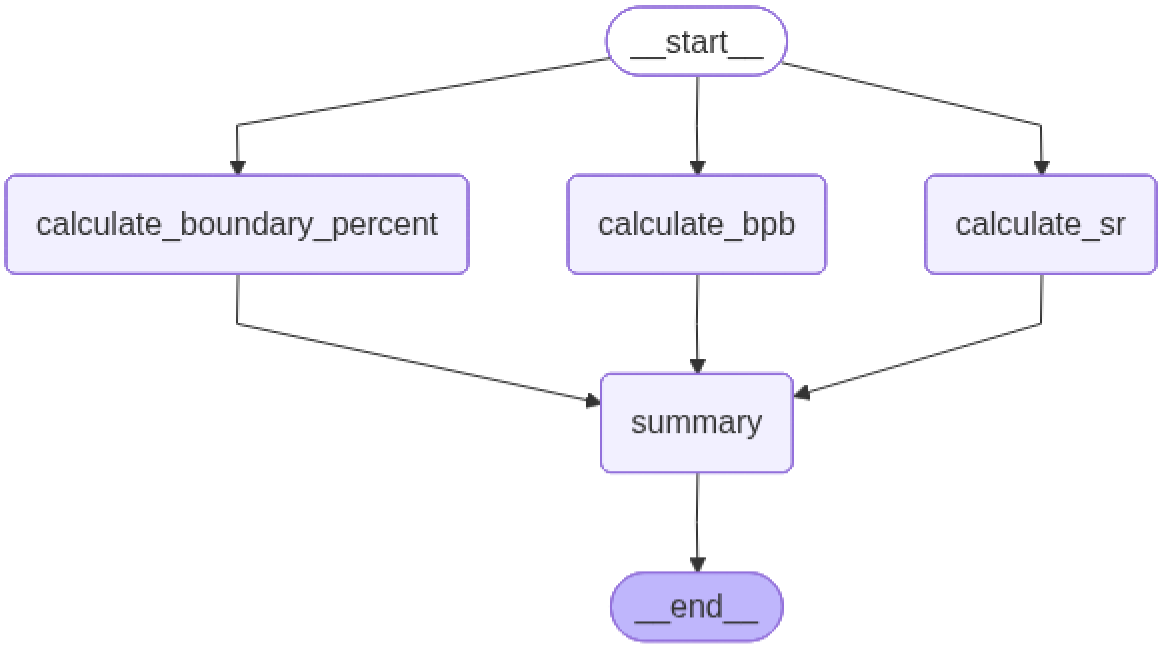
Each node should return:

> Only what it changed

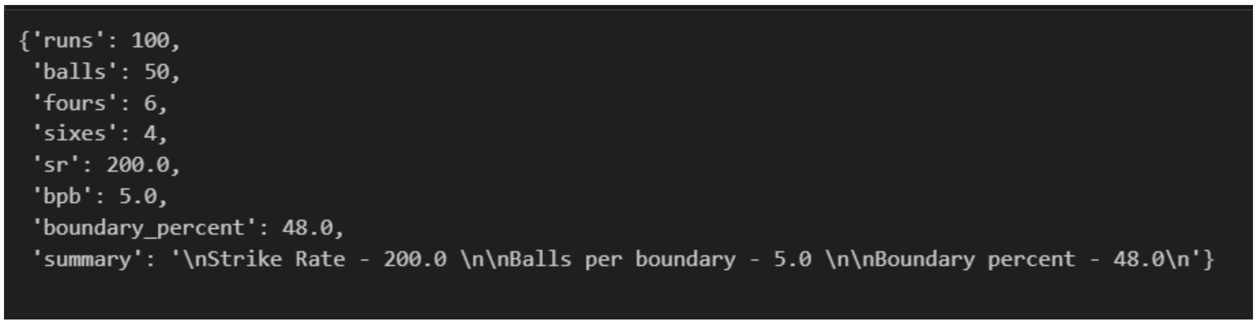
Not the full state.

Edges & Nodes:

```
graph = StateGraph(BatsmanState)

# Nodes

graph.add_node('calculate_sr', calculate_sr)
graph.add_node('calculate_bpb', calculate_bpb)
graph.add_node('calculate_boundary_percent', calculate_boundary_percent)
graph.add_node('summary', summary)

# Edges

graph.add_edge(START, 'calculate_sr')
graph.add_edge(START, 'calculate_bpb')
graph.add_edge(START, 'calculate_boundary_percent')

graph.add_edge('calculate_sr', 'summary')
graph.add_edge('calculate_bpb', 'summary')
graph.add_edge('calculate_boundary_percent', 'summary')

graph.add_edge('summary', END)

workflow = graph.compile()
```



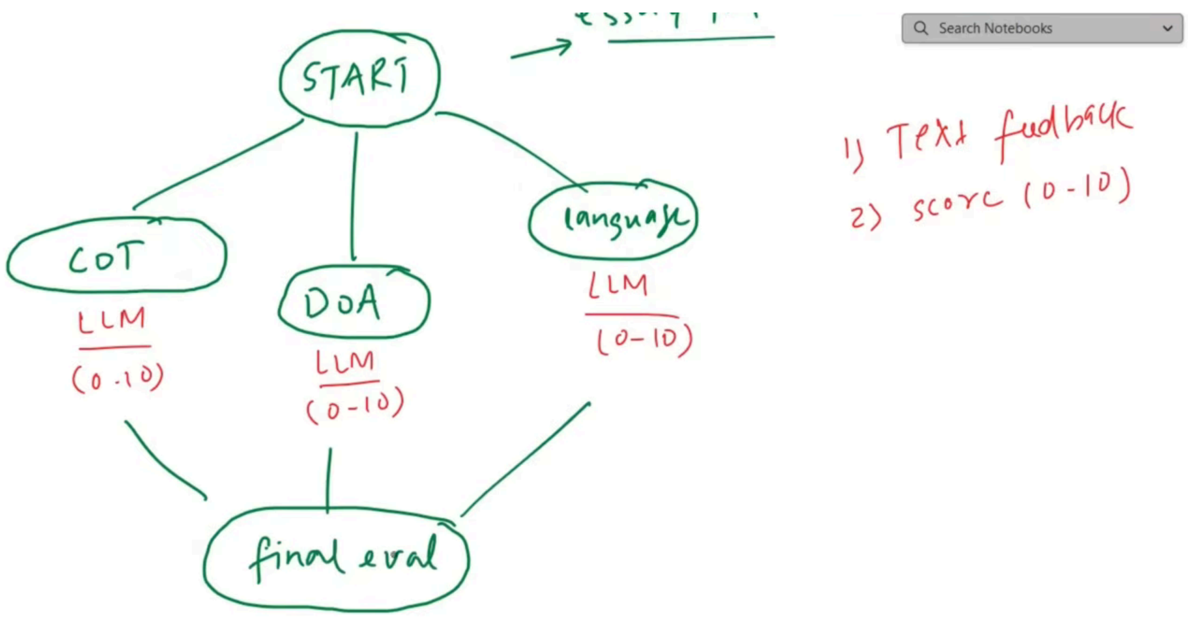
```
intial_state = {
    'runs': 100,
    'balls': 50,
    'fours': 6,
    'sixes': 4
}

workflow.invoke(intial_state)
```



```
{'runs': 100,
 'balls': 50,
 'fours': 6,
 'sixes': 4,
 'sr': 200.0,
 'bpb': 5.0,
 'boundary_percent': 48.0,
 'summary': '\nStrike Rate - 200.0 \n\nBalls per boundary - 5.0 \n\nBoundary percent - 48.0\n'}
```

UPSC Essay Parallel WF:

- Write essay → Evaluate & give feedback



CoT : Clarity of thought

DoA : Depth of Analysis

- Generate summarized feedback
- Return a final average score

💡 **We want structured output:**

1. **Text feedback**
2. **Score (0-10)**

- For this, use → `model.` `with_structured_output` `(EvaluationSchema)`
- Format → JSON

> We need to use **reducer function** as we have to merge multiple scores.

Code:

```
from langgraph.graph import StateGraph, START, END
from langchain_groq import ChatGroq
from typing import TypedDict, Annotated
from pydantic import BaseModel, Field
from dotenv import load_dotenv
import operator
```

`BaseModel, Field` → For structured output validation

`operator` → to add numbers to list

```
model = ChatGroq(model="openai/gpt-oss-20b")
```

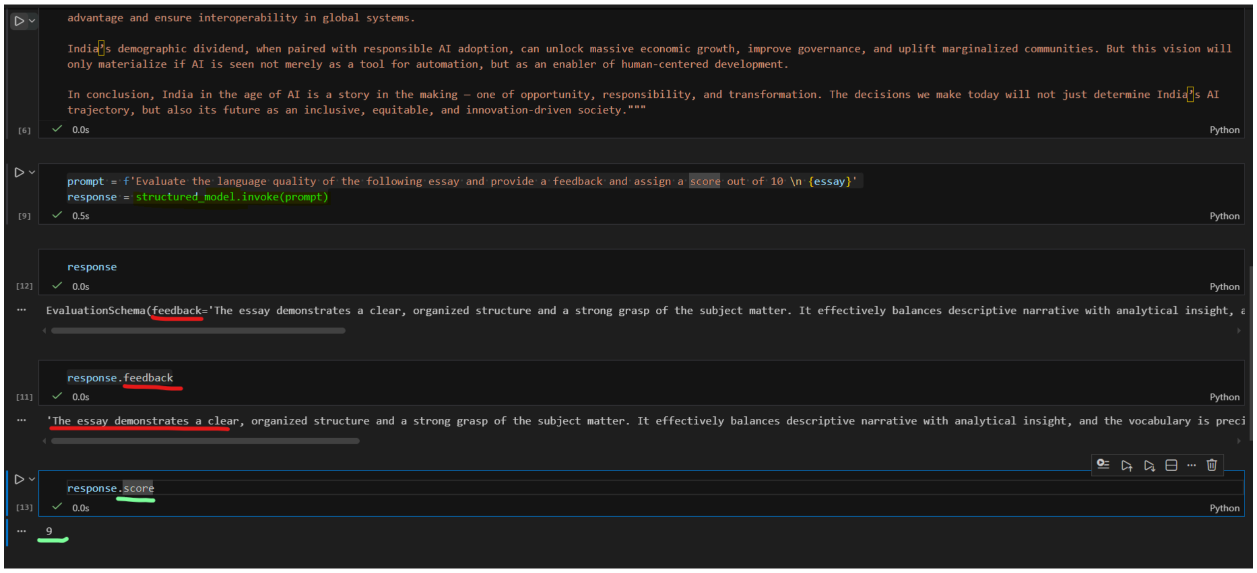
Structured Output:

```
classEvaluationSchema(BaseModel):

    feedback: str = Field(description='Detailed feedbackfor the essay')
    score: int = Field(description='Score out of 10', ge=0, le=10)
```

- Pass the above schema in the function `with_structured_output`

```
structured_model = model.with_structured_output(EvaluationSchema)
```

Test:



```
advantage and ensure interoperability in global systems.

India's demographic dividend, when paired with responsible AI adoption, can unlock massive economic growth, improve governance, and uplift marginalized communities. But this vision will
only materialize if AI is seen not merely as a tool for automation, but as an enabler of human-centered development.

In conclusion, India in the age of AI is a story in the making — one of opportunity, responsibility, and transformation. The decisions we make today will not just determine India's AI
trajectory, but also its future as an inclusive, equitable, and innovation-driven society."""
```
[6] 0.0s Python

```
prompt = f'Evaluate the language quality of the following essay and provide a feedback and assign a score out of 10 \n {essay}'
response = structured_model.invoke(prompt)
```
[9] 0.5s Python

```
response
```
[12] 0.0s Python

```
EvaluationSchema(feedback='The essay demonstrates a clear, organized structure and a strong grasp of the subject matter. It effectively balances descriptive narrative with analytical insight, a
```

```
response.feedback
```
[11] 0.0s Python

```
'The essay demonstrates a clear, organized structure and a strong grasp of the subject matter. It effectively balances descriptive narrative with analytical insight, and the vocabulary is preci
```

```
response.score
```
[13] 0.0s Python

```
9
```

Define state:

```
class UPSCState(TypedDict):

    essay: str
    language_feedback: str
    analysis_feedback: str
    clarity_feedback: str
    overall_feedback: str
    individual_scores: Annotated[list[int], operator.add]
    avg_score: float
```

- We have 3 feedbacks + 1 overall feedback.
- We are storing individual scores in the form of list

`operator.add` → We'll get score in the form of list. (eg. `[8]` , `[9]`)

- To add these scores, we have to write `[8] +[9]`
- But we can't write this in the above state, therefore, we write `operator.add` to add these numbers to a list

Define Functions

```
def evaluate_language(state: UPSCState):

    prompt = f'Evaluate the language quality of the following essay and provide
a feedback and assign a score out of 10 \n {state["essay"]}'
    output = structured_model.invoke(prompt)

    return {'language_feedback': output.feedback, 'individual_scores': [output.s
core]}
```

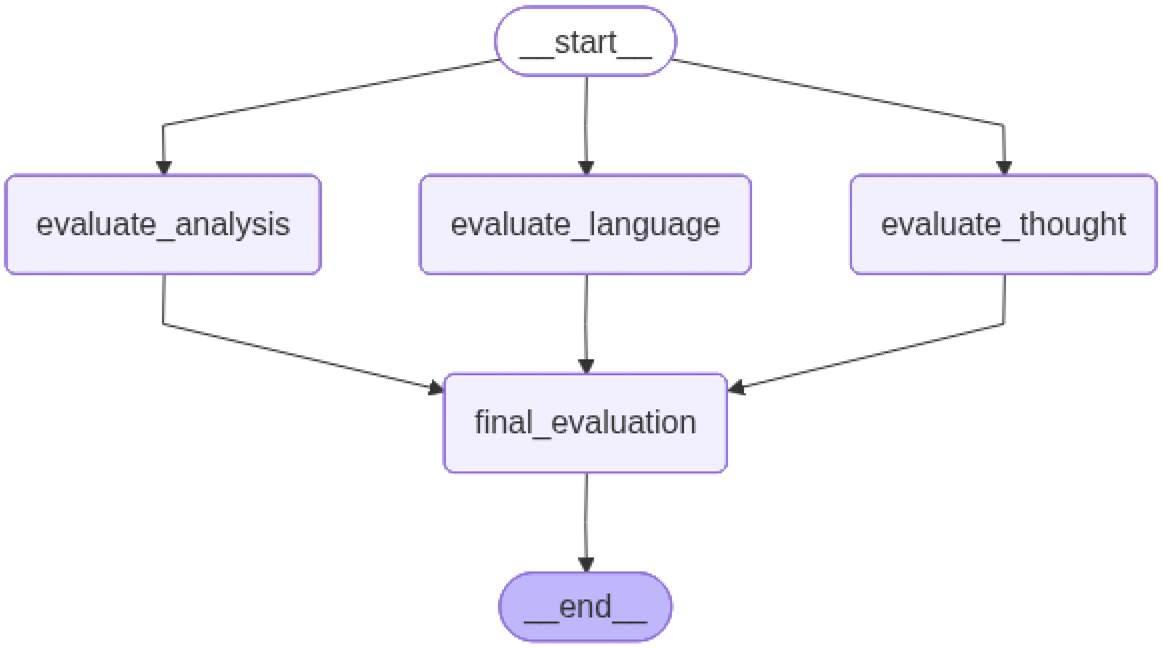
```
def evaluate_analysis(state: UPSCState):

    prompt = f'Evaluate the depth of analysis of the following essay and provid
e a feedback and assign a score out of 10\n{state["essay"]}'
    output = structured_model.invoke(prompt)

    return {'analysis_feedback': output.feedback, 'individual_scores': [output.sc
ore]}
```

```
def evaluate_thought(state: UPSCState):

    prompt = f'Evaluate the clarity of thought of the following essay and provid
e a feedback and assign a score out of 10 \n {state["essay"]}'
    output = structured_model.invoke(prompt)

    return {'clarity_feedback': output.feedback, 'individual_scores': [output.scor
e]}
```

```
deffinal_evaluation(state: UPSCState):

    # summary feedback
    prompt = f'Based on the following feedbacks create a summarized feedbac
k\n language feedback -{state["language_feedback"]}\n depth of analysis fee
dback -{state["analysis_feedback"]}\n clarity of thought feedback -{state["cla
rity_feedback"]}'
    overall_feedback = model.invoke(prompt).content

    # avg calculate
    avg_score = sum(state['individual_scores'])/len(state['individual_scores'])

    return {'overall_feedback': overall_feedback, 'avg_score': avg_score}
```

Add nodes:

```
graph = StateGraph(UPSCState)

graph.add_node('evaluate_language', evaluate_language)
graph.add_node('evaluate_analysis', evaluate_analysis)
graph.add_node('evaluate_thought', evaluate_thought)
graph.add_node('final_evaluation', final_evaluation)

# edges
graph.add_edge(START, 'evaluate_language')
graph.add_edge(START, 'evaluate_analysis')
graph.add_edge(START, 'evaluate_thought')

graph.add_edge('evaluate_language', 'final_evaluation')
graph.add_edge('evaluate_analysis', 'final_evaluation')
graph.add_edge('evaluate_thought', 'final_evaluation')

graph.add_edge('final_evaluation', END)

workflow = graph.compile()
workflow
```

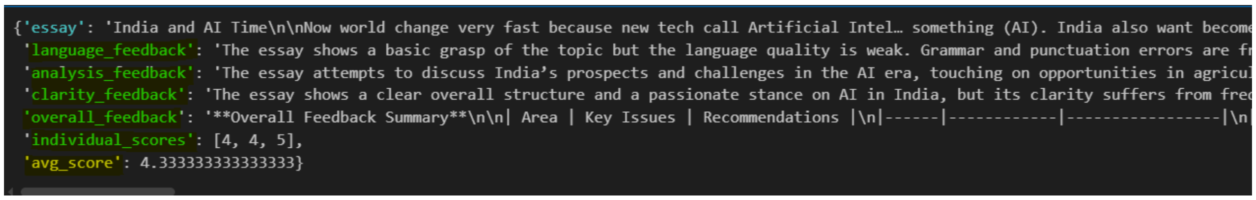


Test

```
essay2 = """India and AI Time

Now world change very fast because new tech call Artificial Intel... something
(AI). India also want become big in this AI thing. If work hard, India can go top.
But if no careful, India go back.

India have many good. We have smart student, many engine-ear, and good IT
peoples. Big company like TCS, Infosys, Wipro already use AI. Government als
o do program “AI for All”. It want AI in farm, doctor place, school and transpor
t.

In farm, AI help farmer know when to put seed, when rain come, how stop bu
g. In health, AI help doctor see sick early. In school, AI help student learn goo
d. Government office use AI to find bad people and work fast.

But problem come also. First is many villager no have phone or internet. So AI
not help them. Second, many people lose job because AI and machine do wor
k. Poor people get more bad.
```

```
One more big problem is privacy. AI need big big data. Who take care? India st
ill make data rule. If no strong rule, AI do bad.

India must all people together – govern, school, company and normal people.
We teach AI and make sure AI not bad. Also talk to other country and learn fro
m them.

If India use AI good way, we become strong, help poor and make better life. B
ut if only rich use AI, and poor no get, then big bad thing happen.

So, in short, AI time in India have many hope and many danger. We must go ri
ght road. AI must help all people, not only some. Then India grow big and worl
d say "good job India"."""
```

```
intial_state = {
    'essay': essay2
}

workflow.invoke(intial_state)
```



```
{'essay': 'India and AI Time\n\nNow world change very fast because new tech call Artificial Intel… something (AI). India also want become
 'language_feedback': 'The essay shows a basic grasp of the topic but the language quality is weak. Grammar and punctuation errors are fr
 'analysis_feedback': 'The essay attempts to discuss India's prospects and challenges in the AI era, touching on opportunities in agricul
 'clarity_feedback': 'The essay shows a clear overall structure and a passionate stance on AI in India, but its clarity suffers from fre
 'overall_feedback': '**Overall Feedback Summary**\n\n| Area | Key Issues | Recommendations |\n|------|-----------|----------------|\n|
 'individual_scores': [4, 4, 5],
 'avg_score': 4.333333333333333}
```